

```

1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Windows.Forms;
9
10 namespace SimpleEvents
11 {
12     public partial class Form1 : Form
13     {
14         Person person = new Person();
15
16         public Form1()
17         {
18             InitializeComponent();
19             person.FirstName = "Christian";
20             person.LastName = "Foss";
21         }
22
23         private void button1_Click(object sender, EventArgs e)
24         {
25             person.HairColor = textBox1.Text;
26         }
27     }
28 }

```

Introducing Scientific Computing

Dave Biersach
dbiersach@bnl.gov

Welcome!

- My name is **Dave Biersach**
- I am a Technology Architect at BNL
- I have been writing software for 35 years
- I am married and have 3 teenage children and 3 dogs
- My college majors were Physics and Math
- Both of my parents were educators
- I wrote all of the code you will see today
- You can email me at dbiersach@bnl.gov

About Brookhaven National Laboratory



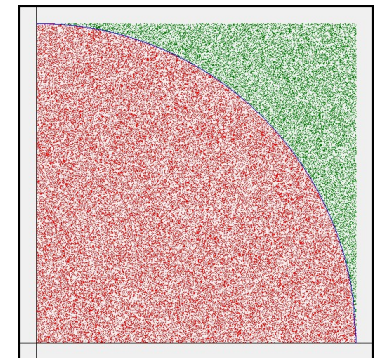
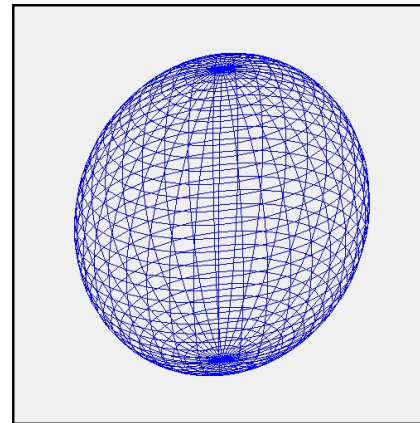
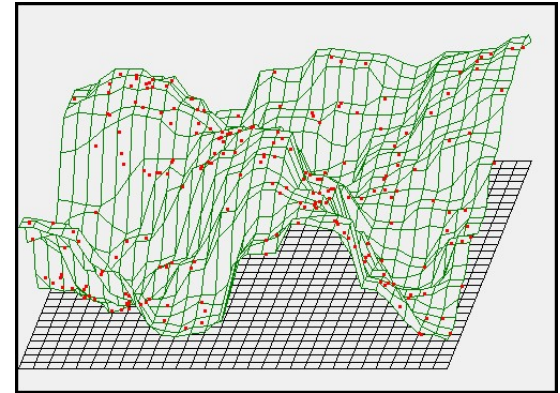
Who We Are

Brookhaven National Laboratory is a multipurpose research institution funded primarily by the U.S. Department of Energy's Office of Science. Located on the center of Long Island, New York, Brookhaven Lab brings world-class facilities and expertise to the most exciting and important questions in basic and applied science—from the birth of our universe to the sustainable energy technology of tomorrow.

We operate cutting-edge large-scale facilities for studies in physics, chemistry, biology, medicine, applied science, and a wide range of advanced technologies. The Laboratory's almost 3,000 scientists, engineers, and support staff are joined each year by more than 4,000 visiting researchers from around the world. Our award-winning history stretches back to 1947, and we continue to unravel mysteries from the nanoscale to the cosmic scale, and everything in between.

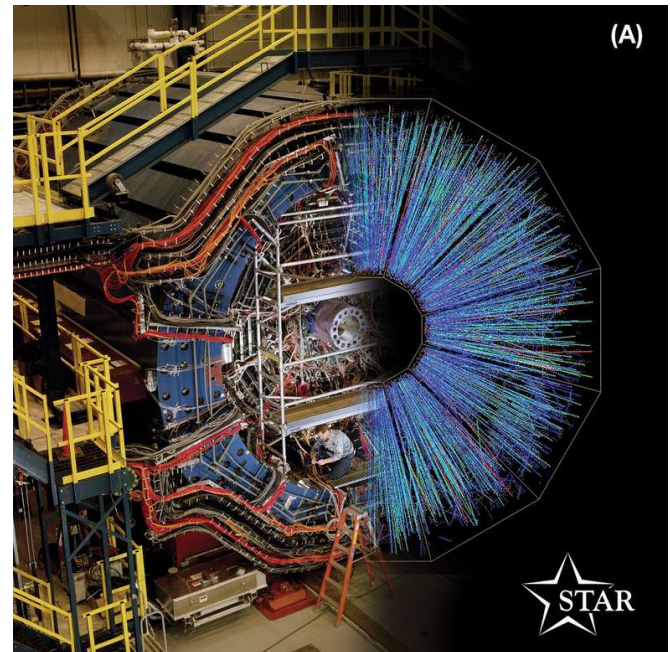
Agenda

- SciComp in Amazon's Cloud
- Rolling for Hero Abilities
- Uniform Magic
- Hunting for Red October
- k-Means Clustering
- Circus Cannon
- Draw Your Own Death Star
- The Longest Gene
- Monte Carlo Integration
- The Right Way to Shuffle
- BNL Initiatives in SciComp Education



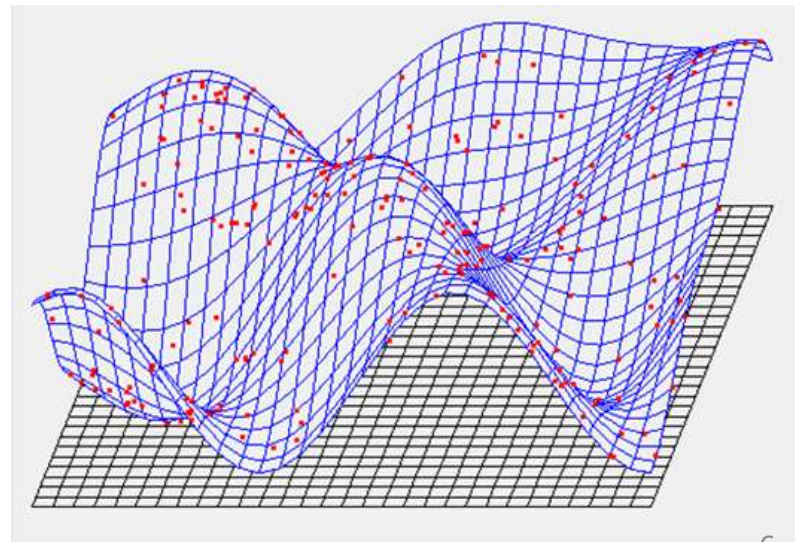
Overall Goals

- Stimulate your curiosity in what is possible
- Introduce a broad range of terms and concepts
- Show how complex programs use the same basic Legos
- Solve science problems by writing custom code
- Allow you to continue learning at home via the cloud



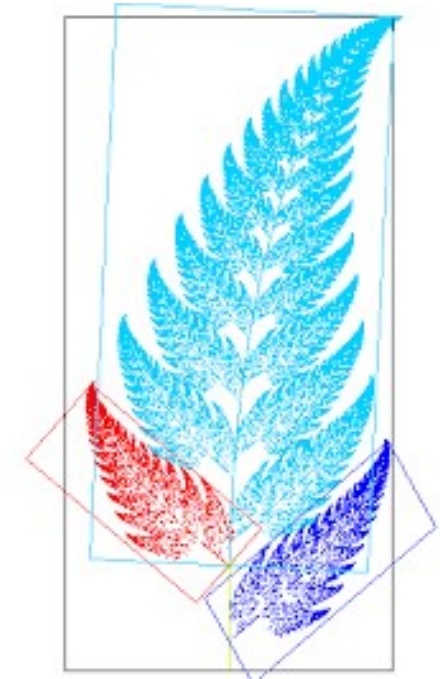
Mathematical Concepts

- Matrices and Systems of Linear Equations
- Probability Distributions (Normal & Uniform)
- Monte Carlo Numerical Integration
- Polynomial Root Finding
- Polar & Spherical Coordinates
- Projectile Motion
- 2D Affine Transformations
- Continued Fractions
- Mesh Interpolation
- Cluster Analysis



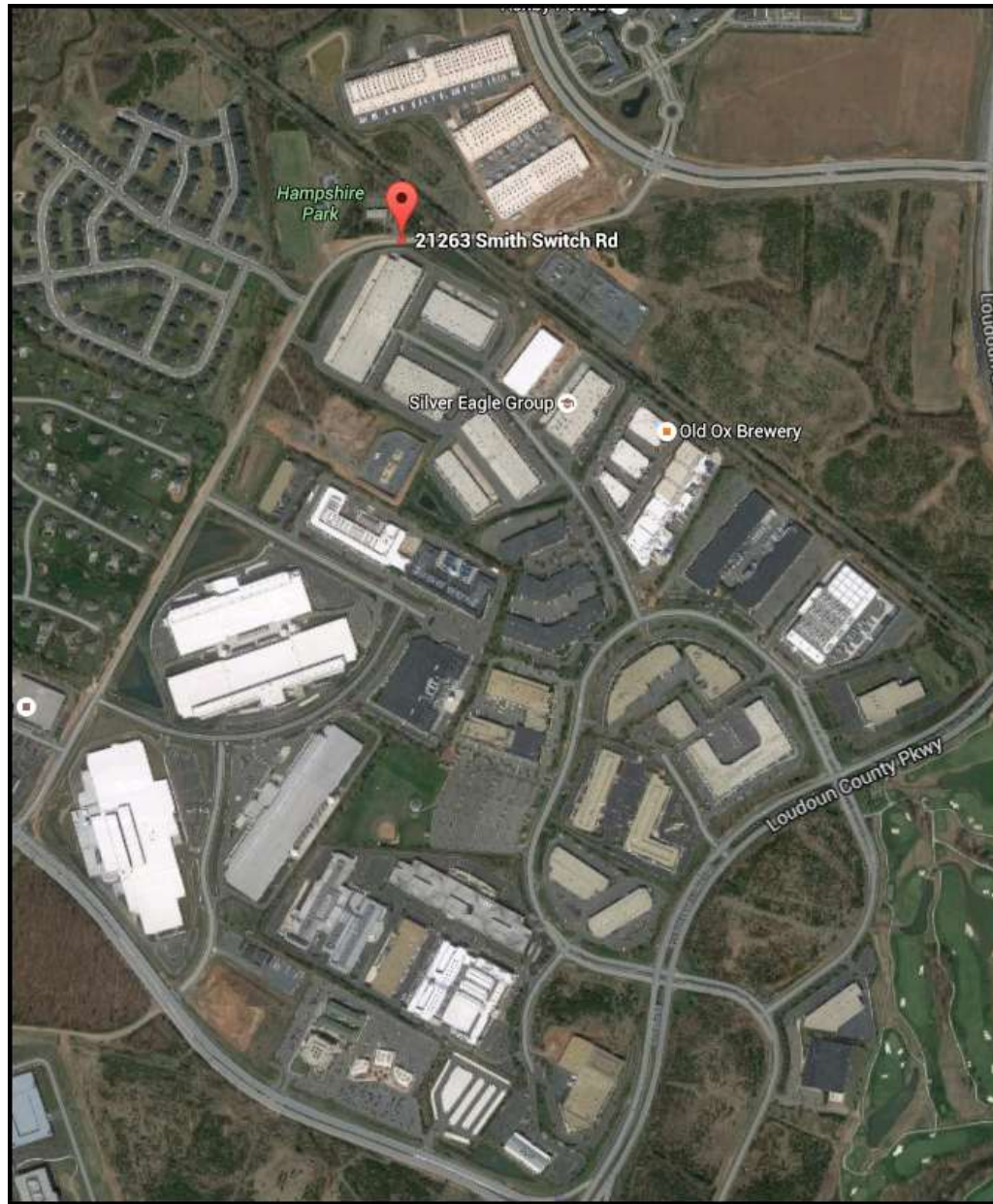
Computer Science Concepts

- Representations and Encodings
- Random Number Generation
- Strings, Arrays, Operators
- Loops, Functions, Recursion
- Sorting (Bubble vs. Quick Sort)
- Searching (Depth vs. Breadth)
- 2D and 2.5D Graphics (Isometric Projection)
- Fractal Image Compression
- Divide and Conquer Algorithms and Runtime Complexity
- Combinations and Permutations



How can we get a computer to...

- Calculate the square root of a number with 1,000 digits?
- Find the shortest path through a million cell maze?
- Quickly deal a billion card decks?
- Factor a quadratic polynomial with very large coefficients?
- Sort a trillion numbers in a few milliseconds?
- Decipher encrypted files using only statistical analysis?
- Interpolate multi-dimensional scientific data?
- Analyze dynamical systems as they diverge into chaos?
- Thwart hackers attempting to break into BNL web sites?



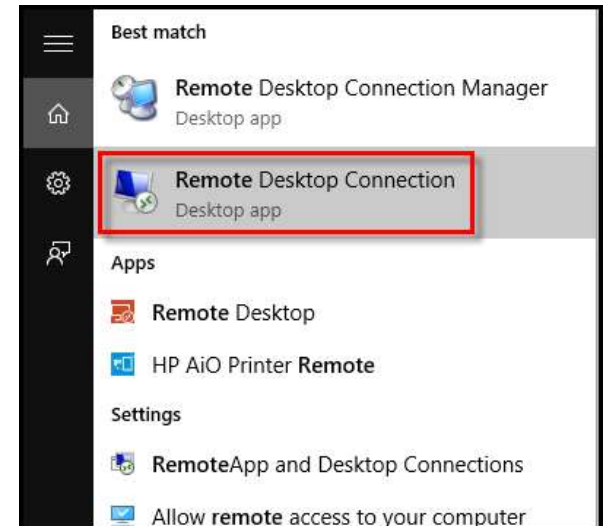
Amazon's
23 data centers in
Ashburn, Virginia

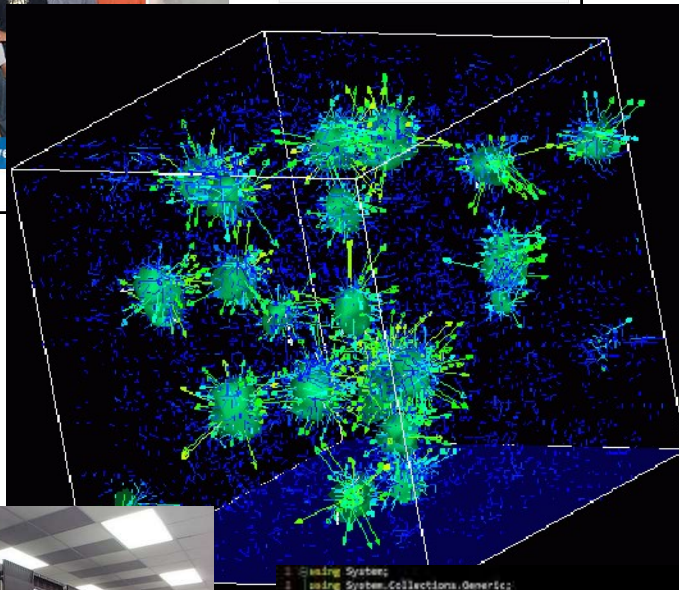
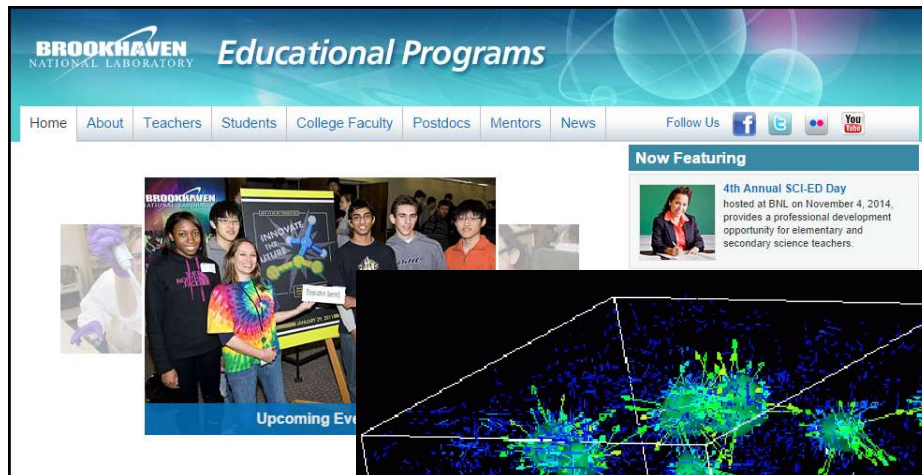
Just two of Amazon's 23 cloud data centers in Ashburn, Virginia



Accessing your Remote PC

- Launch Microsoft's “**Remote Desktop Connection**” application
- Make sure you write down
 - The IP address of your specific machine in Amazon's cloud
 - The user name (case insensitive)
 - The password (case sensitive)
- You can access your Amazon machine from your home computer too!





```

1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Windows.Forms;
9
10 namespace SimpleEvents
11 {
12     public partial class Form1 : Form
13     {
14         Person person = new Person();
15
16         public Form1()
17         {
18             InitializeComponent();
19             person.FirstName = "Christian";
20             person.LastName = "Foss";
21         }
22
23         private void button1_Click(object sender, EventArgs e)
24         {
25             person.HairColor = textBox1.Text;
26         }
27     }
28 }

```

Introducing Scientific Computing

Part 1 Hero Abilities

Generating Hero Ability Values

- In most role-playing games, heroes have abilities such as strength, dexterity, intelligence, charisma, etc.
- Initial abilities are often measured in ranges like **3 – 18**
- At the beginning of the game, players *roll dice* to determine the initial values for each ability
- The higher the value, the more likely the player will succeed while adventuring



| | ABILITY SCORE | ABILITY MODIFIER | TEMP SCORE | TEMP MODIFIER |
|-----|------------------|---------------------|---------------|------------------|
| STR | | | | |
| DEX | | | | |
| CON | | | | |
| INT | | | | |
| WIS | | | | |
| CHA | | | | |

Generating Hero Ability Values

- Two ways of rolling for initial abilities between **3 and 18**:
 - Roll a **20**-sided die just once (**1d20**), but *reroll* if face value is 1, 2, 19, or 20
 - Roll a **6**-sided die three times (**3d6**), summing the value of each roll
- Using the **1d20** method is faster than **3d6**, especially when having to roll for six separate abilities

| | ABILITY SCORE | ABILITY MODIFIER | TEMP SCORE | TEMP MODIFIER |
|-----|------------------|---------------------|---------------|------------------|
| STR | 17 | | | |
| DEX | 11 | | | |
| CON | 15 | | | |
| INT | 5 | | | |
| WIS | 7 | | | |
| CHA | 3 | | | |



Generating Hero Ability Values

- Two ways of rolling for initial abilities between **3 and 18**:
 - Roll a **20**-sided die just once (**1d20**), but *reroll* if face value is 1, 2, 19, or 20
 - Roll a **6**-sided die three times (**3d6**), summing the value of each roll
- Which method would you want to use? Why?

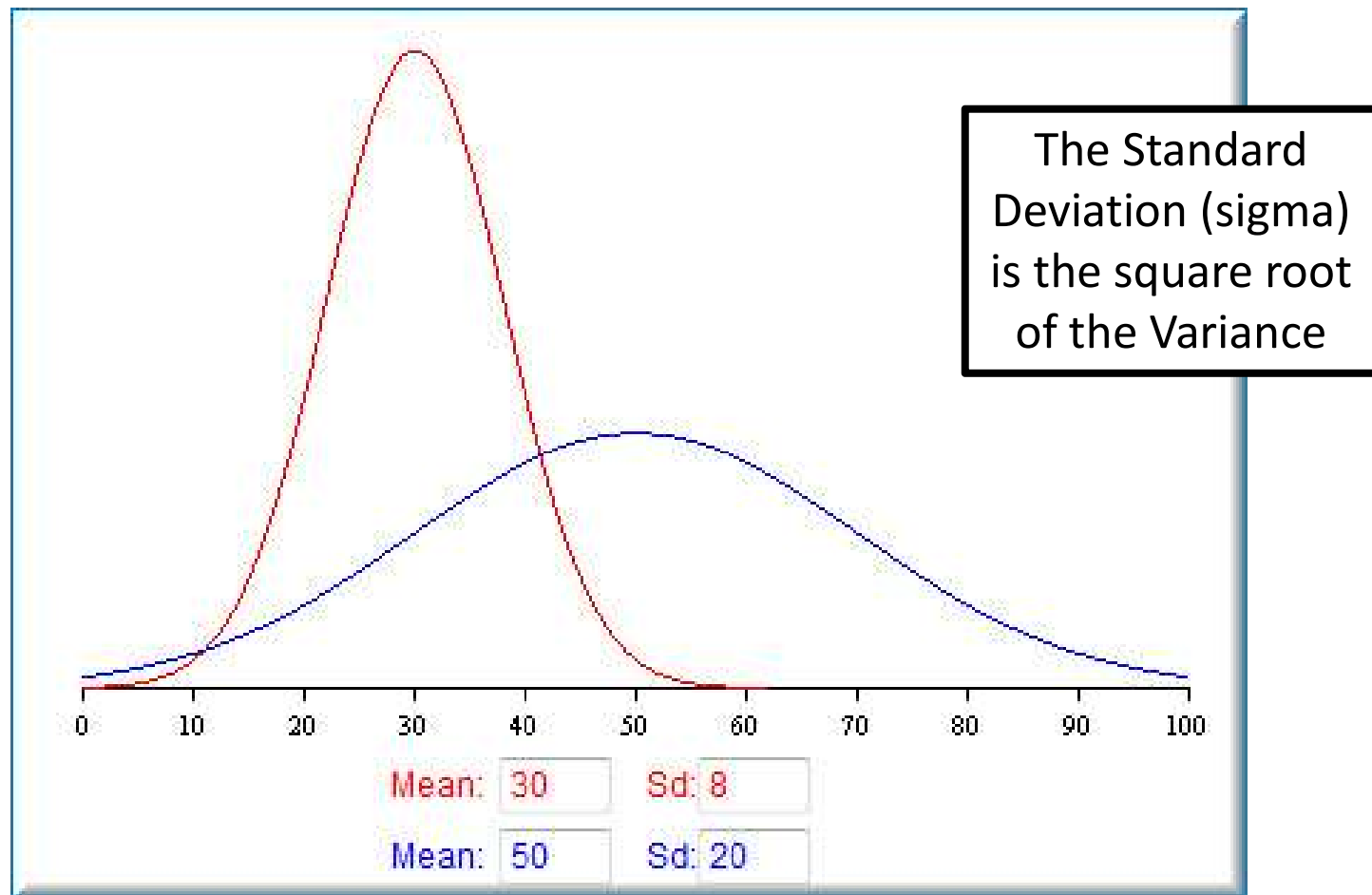
| | ABILITY SCORE | ABILITY MODIFIER | TEMP SCORE | TEMP MODIFIER |
|-----|---------------|------------------|------------|---------------|
| STR | 17 | | | |
| DEX | 11 | | | |
| CON | 15 | | | |
| INT | 5 | | | |
| WIS | 7 | | | |
| CHA | 3 | | | |



Mean vs. Variance

- Imagine two different classes (each with 20 students) taking the exact same chapter test...
 - All Class #1 students score between 70 and 80 \therefore mean = 75
 - All Class #2 students score between 50 and 100 \therefore mean = 75
- **Variance** is the average “distance” between each number in a set and the mean of that set
 - Class #2 scores had a greater **variance** in scores than Class #1
 - Variance is a measure of **central tendency** – on average how close around the mean do all the numbers fall?
 - For every data point, we sum the **square** of the difference between the number and the mean. Then we divide that sum by the total number of data points.

Mean vs. Standard Deviation



Mean, Variance, Standard Deviation

Variance

Mean

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2 \quad \text{Where} \quad \mu = \frac{1}{n} \sum_{i=1}^n x_i$$

$$\sigma = \sqrt{\sigma^2} = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$$

Standard Deviation

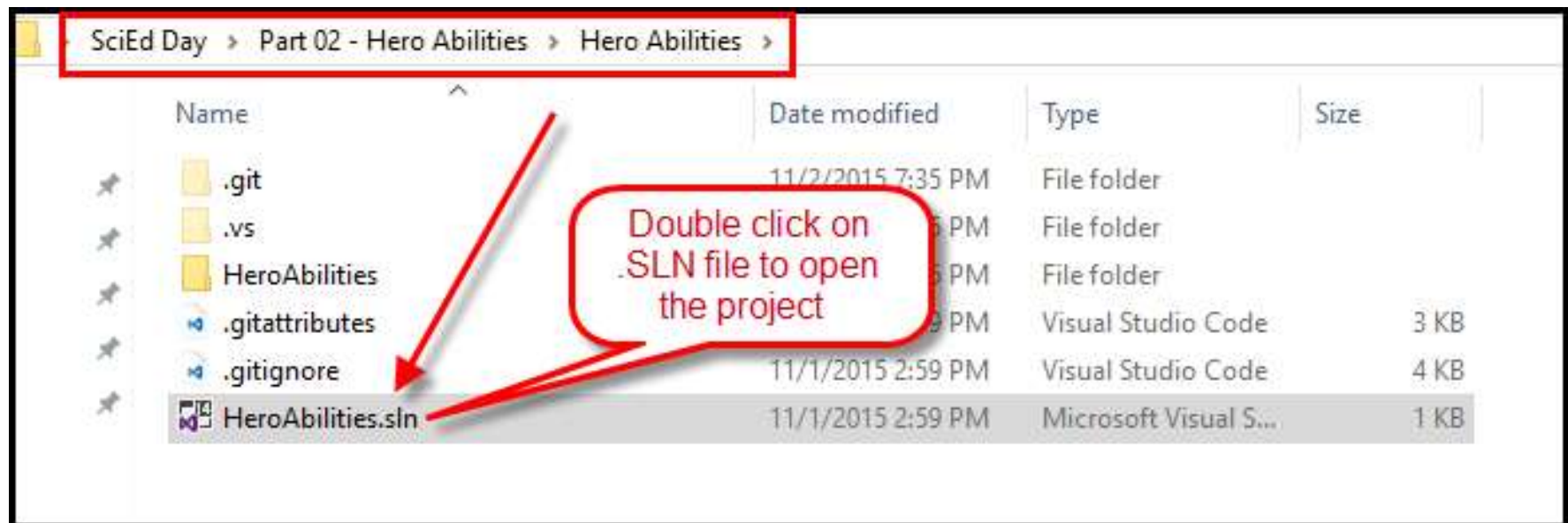
“Random” Numbers

- The built-in **Random** class creates pseudo-random numbers
 - It generates a nearly perfect **uniform distribution** – every number has an equal chance of getting picked at random
 - The **Next**(**int** a, **int** b) method returns a random integer $\geq a$ and $< b$
 - Example: **Next**(1, 11) returns a random integer between 1 and **10**
- When you declare a new instance of a Random class, you must specify the initial seed value
 - If you initialize Random to the same seed value, it will emit the same sequence of numbers **every** time your program runs
 - ***All of our computers*** will return the exact same sequence if we all initialize our PRNG with the same seed value!
 - We will always use the seed value of **2015**

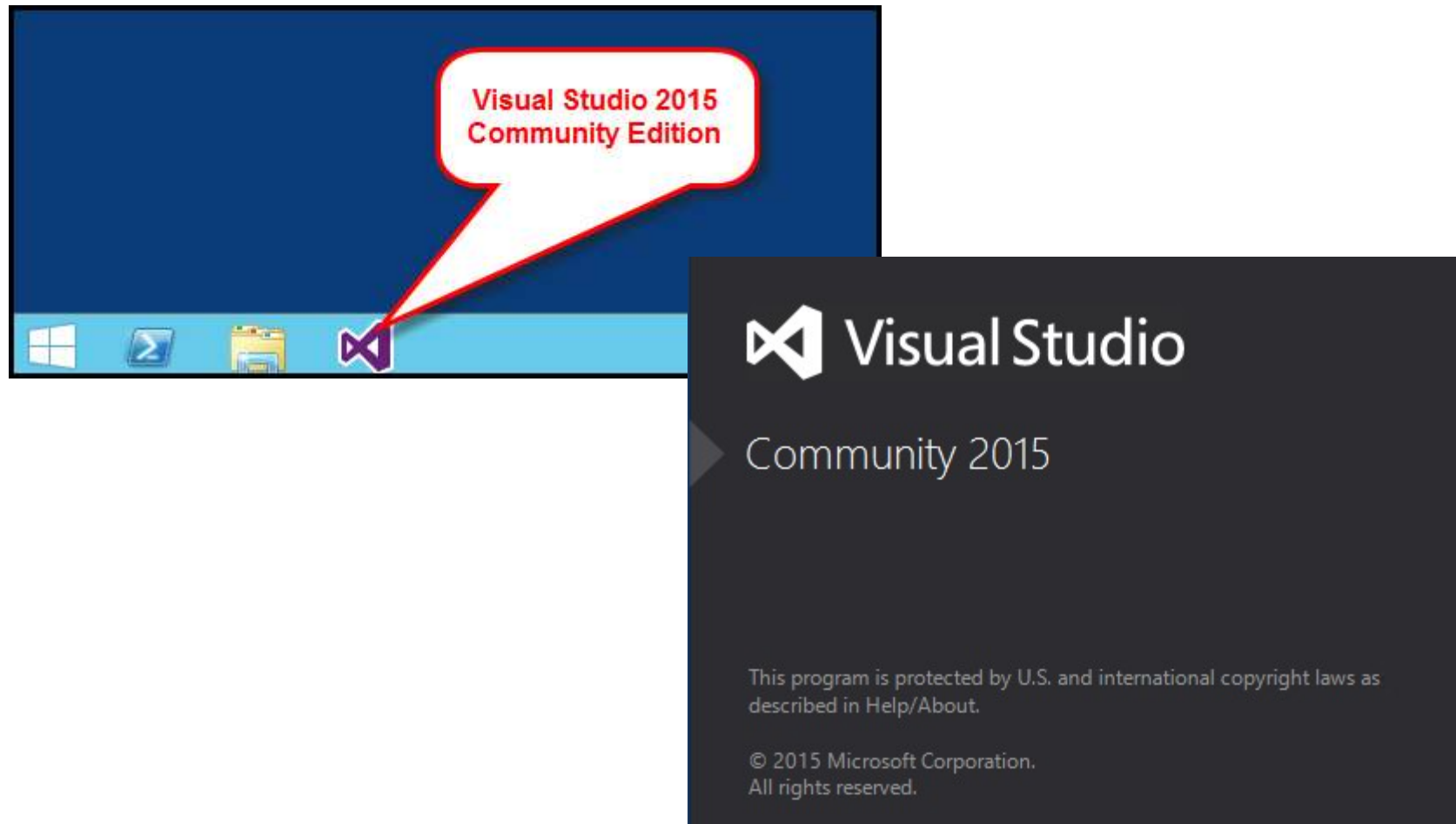
Lab 1

- Difficulty **Level 100**
- Write a program to generate **1,000,000** hero ability scores, comparing the *mean* and *standard deviation* of the 1d20 versus the 3d6 dice roll methods
- In particular, write the missing mathematical expression to correctly define the **stdDev** variable in the **CalcStdDev1d20** function. *Hint: see slide #10*
- **Math.Pow**(x, y) means x^y and **Math.Sqrt**(x) means \sqrt{x}
- Which dice roll method would you want to use to generate your hero's abilities?

Lab 1



Microsoft Visual Studio (C#)



Lab 1

```
static double CalcStdDev1d20(double totalRolls, double mean)
{
    Random rnd = new Random(2015);
    double sum = 0;
    for (double rollNumber = 0; rollNumber < totalRolls; rollNumber++)
    {
        double roll = rnd.Next(3, 19);
        sum = sum + Math.Pow(roll - mean, 2);
    }
    double stdDev = Math.Sqrt(sum / totalRolls);
    return stdDev;
}
```

**This function receives
two inbound values,
both of type double**

Lab 1

```
file:///D:/DaveB/SciComp/SciComp Seminars - Instructor/Seminar 03 - Probability, Conditionals, Functions/Lab 1 - Hero ...
Total number of dice rolls: 1,000,000
Mean ability (1d20): 10.49
Mean ability (3d6) : 10.50
Standard deviation ability (1d20): 4.61
Standard deviation ability (3d6) : 2.96
Press any key to continue...
_
```

Which roll type will most likely create an ability score closest to the mean?

Greatest Common Divisor (GCD)

Example: **What is the GCD of 231 and 182?** In step 0, **A** is always greater than or equal to **B**. In steps 1 and beyond, the **A** value is the *greater* of the prior step's **B** or (**A**-**B**) values. The **B** value is the *lesser* of either the prior step's **B** or (**A** - **B**) values. The algorithm stops when $A - B = 0$, and the GCD was the very last **B** value. Follow along with each step in the table below:

| Finding the GCD of 231 and 182 | | | |
|--------------------------------|-----|-----|-------|
| Step | A | B | A - B |
| 0 | 231 | 182 | 49 |
| 1 | 182 | 49 | 133 |
| 2 | 133 | 49 | 84 |
| 3 | 84 | 49 | 35 |
| 4 | 49 | 35 | 14 |
| 5 | 35 | 14 | 21 |
| 6 | 21 | 14 | 7 |
| 7 | 14 | 7 | 7 |
| 8 | 7 | 7 | 0 |

You can determine
the GCD without
having to **factor**
either of the two
integers!

Lab 2

- Difficulty **Level 200**
- Open the lab 2 solution and add the code to calculate the average number of times a **million** pairs of random integers ($1 \leq n < 100,000$) are coprime ($\text{GCD} == 1$)
- What does this experiment estimate to be the probability that two randomly chosen integers are coprime?
- **Divide 6 by this average, then take the square root** - what universal constant is lurking there?

Lab 2

```
for (double i = 0; i < maxIterations; i++)  
{  
    int a = r.Next(1, 100000);  
    int b = r.Next(1, 100000);  
  
    if (GCD(a,b) == 1)  
    {  
        coprimePairs = coprimePairs + 1;  
    }  
}
```

**Use the double
equals == operator
to test for equality**

Lab 2

```
file:///D:/DaveB/SciComp/SciComp Seminars - Instructor/Seminar 03 - Probability, Conditionals, Functions/Lab 2 - Copri...
Probability two random integers are coprime = 60.79 %
Hidden constant: 3.141751
Archimedes' constant: 3.141593
Press any key to continue...
```

$$\prod_{\text{prime } p} \left(1 - \frac{1}{p^2}\right) = \left(\prod_{\text{prime } p} \frac{1}{1 - p^{-2}}\right)^{-1} = \frac{1}{\zeta(2)} = \frac{6}{\pi^2} \approx 0.607927102 \approx 61\%.$$


Lab 3



- Difficulty **Level 300**
- Write a program to perform a **million runs** of an experiment that places a varying number of straws *end-to-end* each run
- In each run, start with a single straw of random length between $0 \leq n < 1$
- Then enter a loop that keeps adding additional straws of random length ($0 \leq n < 1$) until the total length is > 1
- Find the **mean** number of straws added before length becomes > 1 , across all million runs of the experiment

Lab 3



Straws

1



2



3



1



2



1



2



3



4



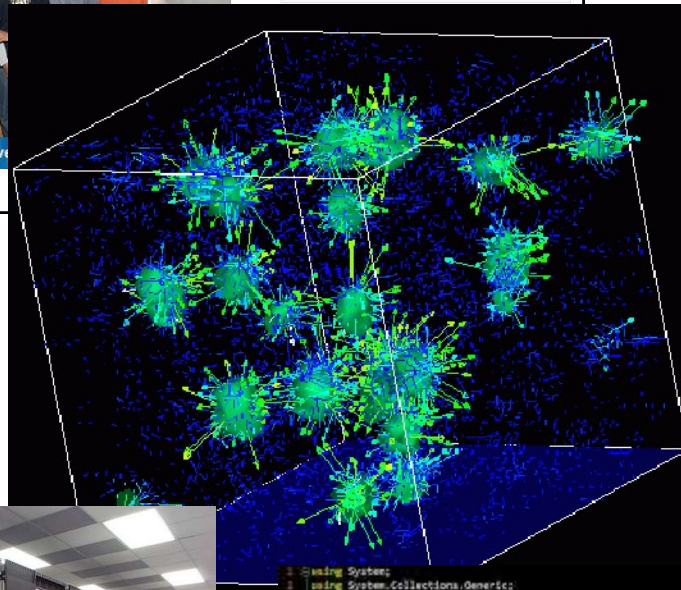
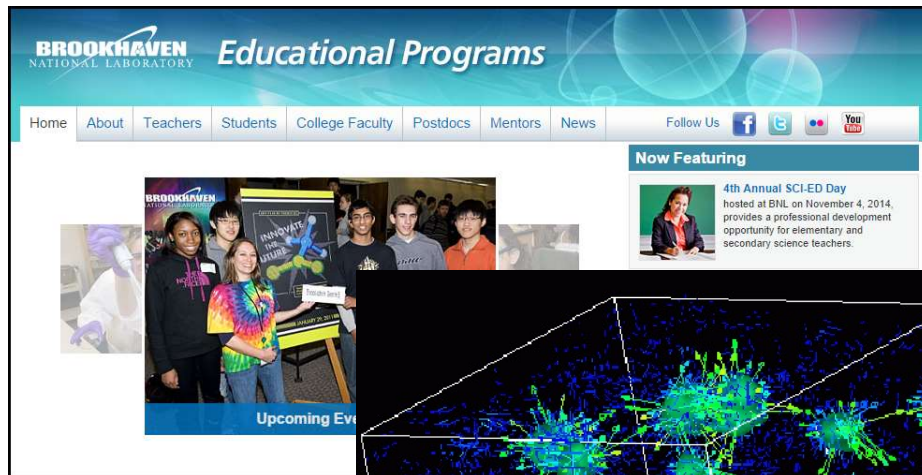
Lab 3

```
file:///D:/DaveB/SciComp/SciComp Seminars/Seminar 03 - Probability, Conditionals, Functions/Bonus 1 - Random Straw...  
Mean straws per iteration: 2.7182763  
Base of natural logarithm: 2.7182818  
Press any key to continue...
```

$$e = \lim_{x \rightarrow \infty} \left(1 + \frac{1}{x}\right)^x = 2.718281828459...$$

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \dots$$





```

1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Windows.Forms;
9
10 namespace SimpleEvents
11 {
12     public partial class Form1 : Form
13     {
14         Person person = new Person();
15
16         public Form1()
17         {
18             InitializeComponent();
19             person.FirstName = "Christian";
20             person.LastName = "Foss";
21         }
22
23         private void button1_Click(object sender, EventArgs e)
24         {
25             person.HairColor = textBox1.Text;
26         }
27     }
28 }

```

Introducing Scientific Computing

Part 2 Uniform Magic

Lab

- Write a program to discover a magic number hidden in all uniform random number distributions
 - Generate 15 sets of random size between 1 million & 2 million items
 - Within each set, every item is a random integer chosen within a range between a **lower limit** and an **upper limit**
 - The **lower limit** is a random number between 0 and 1000
 - The **upper limit** is the sum of the lower limit **and** another random number between 0 and 1000
 - Calculate the mean (μ) and variance (σ^2) for each set

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2 \quad \text{Where} \quad \mu = \frac{1}{n} \sum_{i=1}^n x_i$$

Lab

- Write a program to discover a magic number hidden in all uniform random number distributions
 - Calculate and display this *magic number* for each set:

$$\frac{(\textit{Upper Limit} - \textit{Lower Limit})^2}{\textit{Variance}}$$

- This magic number is the same for ALL uniform distributions!
- You can prove a die is loaded if you do not get this magic number after about 20 rolls!




Lab

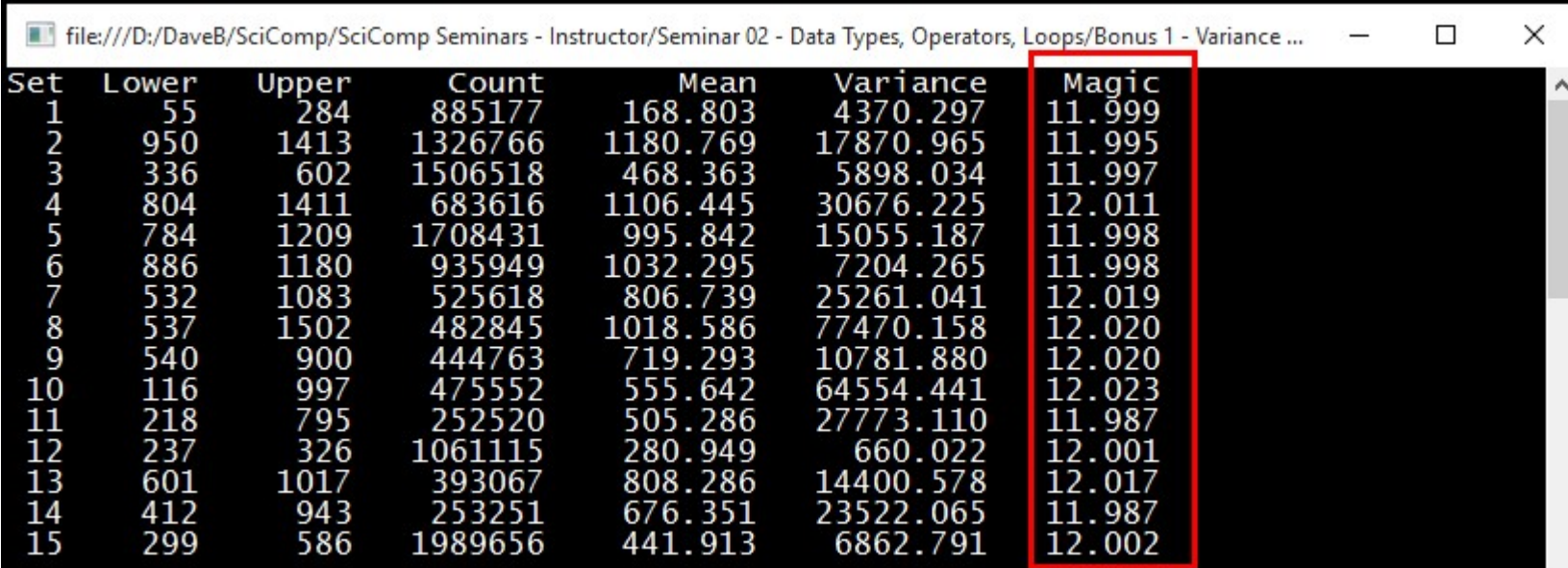
```
// Now find the variance of this set
double sumVariance = 0;
for (double iteration = 0; iteration < totalIterations; iteration++)
{
    int randomNumber = r2.Next(lowerLimit, upperLimit);
    sumVariance = sumVariance + (randomNumber - mean) * (randomNumber - mean);
}
double variance = sumVariance / totalIterations;

// All uniform distributions have this same magic number!
double magicNumber = 0;

// Display the set #, mean, variance, and magic number
Console.Write($"{setNumber,3:D0} {lowerLimit,5:D0} {upperLimit,5:D0} {totalIterations,7:F0}");
Console.WriteLine($" {mean,8:F3} {variance,9:F3} {magicNumber, 5:F3}");
```



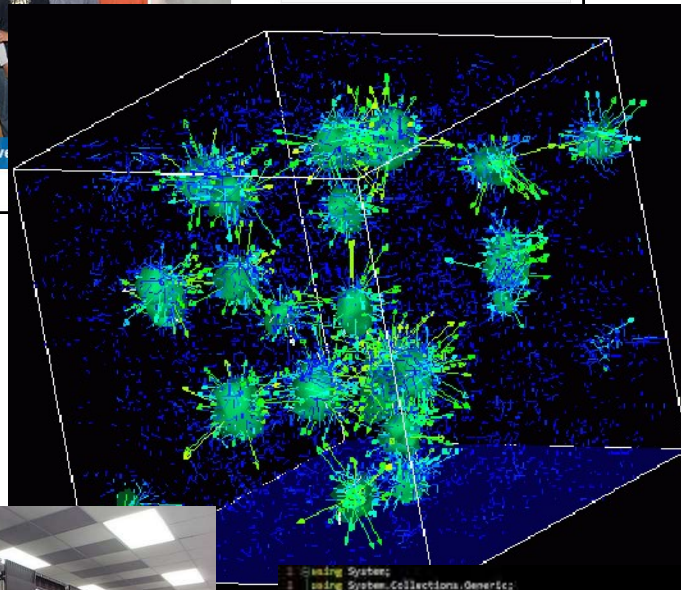
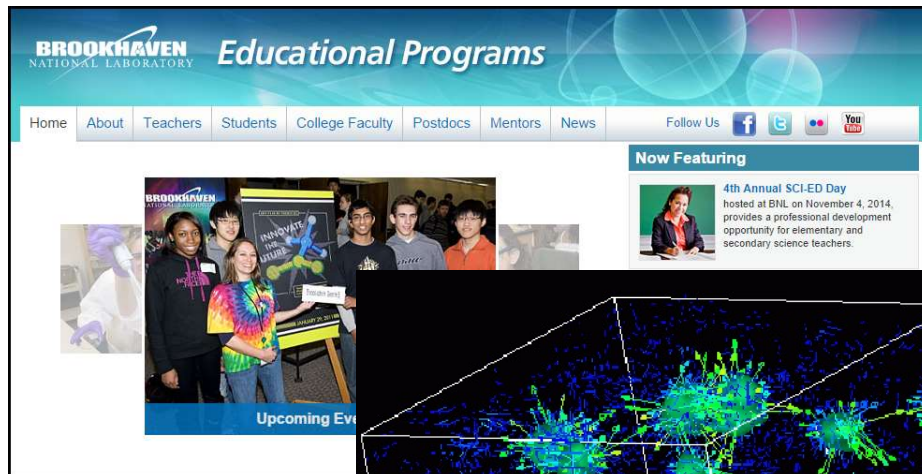
Lab



file:///D:/DaveB/SciComp/SciComp Seminars - Instructor/Seminar 02 - Data Types, Operators, Loops/Bonus 1 - Variance ...

| Set | Lower | Upper | Count | Mean | Variance | Magic |
|-----|-------|-------|---------|----------|-----------|--------|
| 1 | 55 | 284 | 885177 | 168.803 | 4370.297 | 11.999 |
| 2 | 950 | 1413 | 1326766 | 1180.769 | 17870.965 | 11.995 |
| 3 | 336 | 602 | 1506518 | 468.363 | 5898.034 | 11.997 |
| 4 | 804 | 1411 | 683616 | 1106.445 | 30676.225 | 12.011 |
| 5 | 784 | 1209 | 1708431 | 995.842 | 15055.187 | 11.998 |
| 6 | 886 | 1180 | 935949 | 1032.295 | 7204.265 | 11.998 |
| 7 | 532 | 1083 | 525618 | 806.739 | 25261.041 | 12.019 |
| 8 | 537 | 1502 | 482845 | 1018.586 | 77470.158 | 12.020 |
| 9 | 540 | 900 | 444763 | 719.293 | 10781.880 | 12.020 |
| 10 | 116 | 997 | 475552 | 555.642 | 64554.441 | 12.023 |
| 11 | 218 | 795 | 252520 | 505.286 | 27773.110 | 11.987 |
| 12 | 237 | 326 | 1061115 | 280.949 | 660.022 | 12.001 |
| 13 | 601 | 1017 | 393067 | 808.286 | 14400.578 | 12.017 |
| 14 | 412 | 943 | 253251 | 676.351 | 23522.065 | 11.987 |
| 15 | 299 | 586 | 1989656 | 441.913 | 6862.791 | 12.002 |

- Every set had a different lower and upper limit, size, mean, and variance... yet the magic number was **12** for all of them!
- **Why** would Mother Nature pick the value 12 for this magic number? What is so special about 12? Why not pick a nice even 10?
- Boundless natural curiosity is what makes a good scientist...



```

1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Windows.Forms;
9
10 namespace SimpleEvents
11 {
12     public partial class Form1 : Form
13     {
14         Person person = new Person();
15
16         public Form1()
17         {
18             InitializeComponent();
19             person.FirstName = "Christian";
20             person.LastName = "Foss";
21         }
22
23         private void button1_Click(object sender, EventArgs e)
24         {
25             person.HairColor = textBox1.Text;
26         }
27     }
28 }

```

Introducing Scientific Computing

Part 3 Hunting for Red October

Interpolating Spatial Data

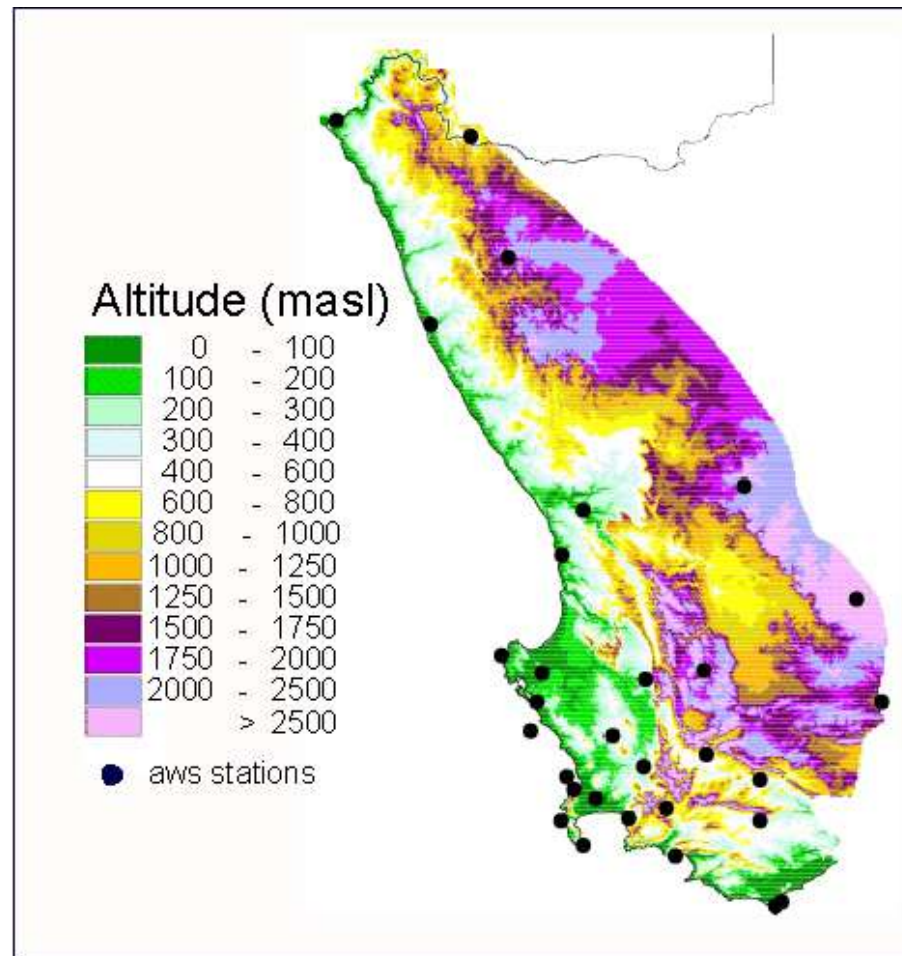
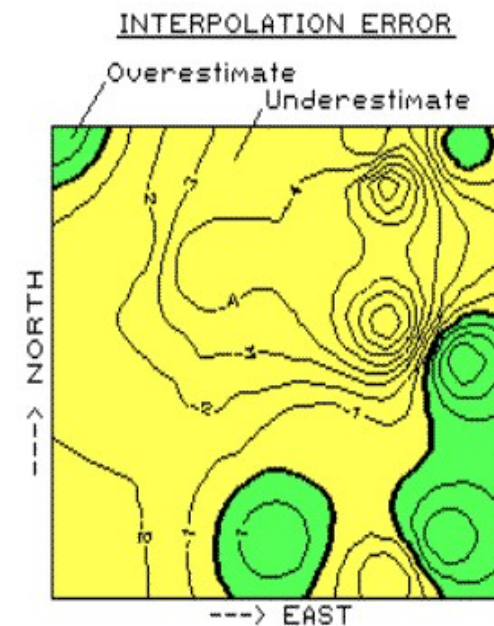
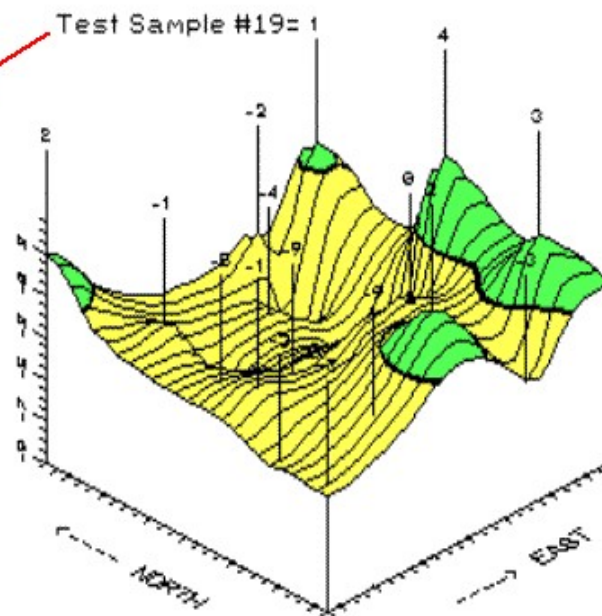


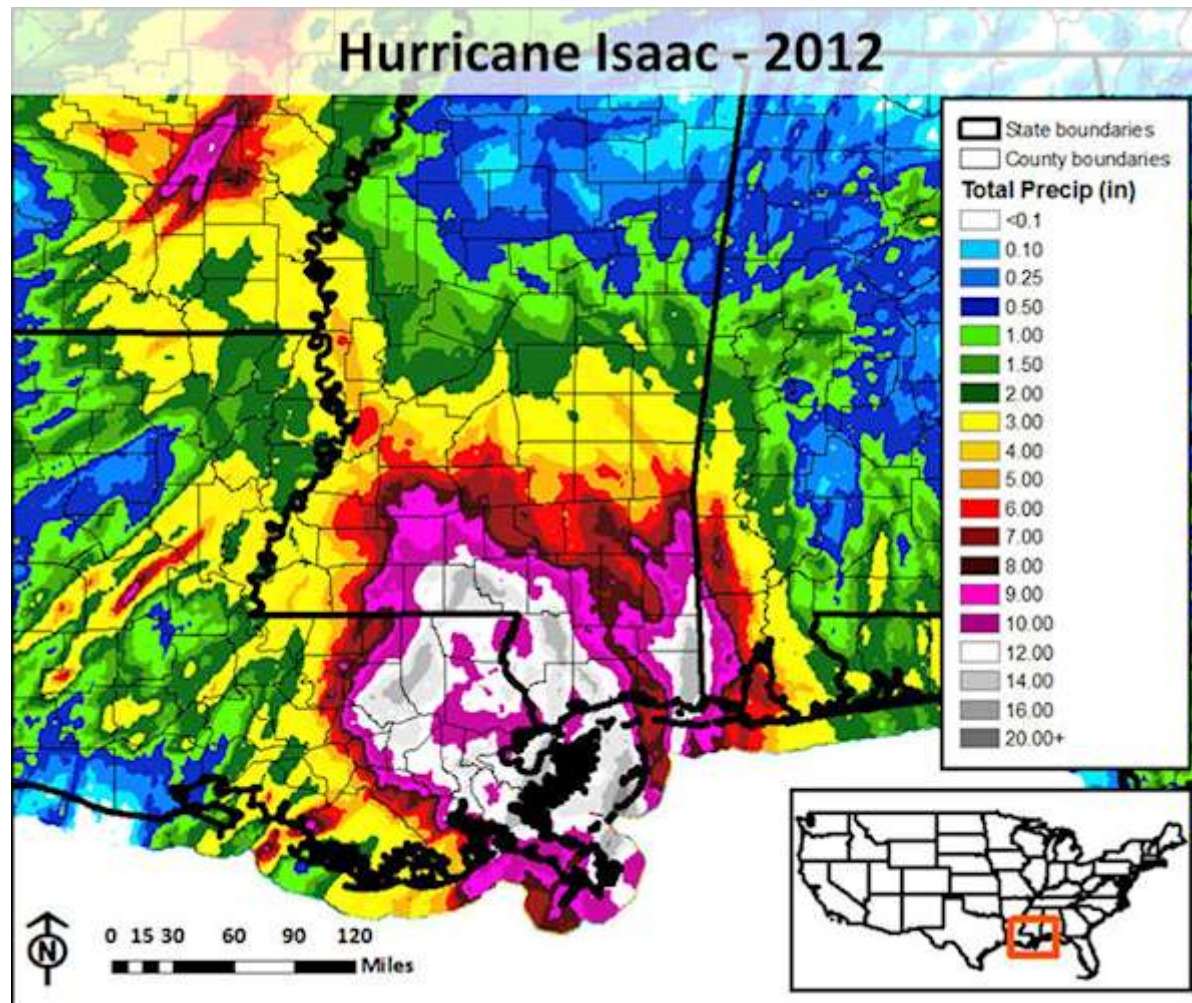
Figure 6 Altitude (200x200m) over the winter rainfall region of South Africa (after Directorate of Land Surveys and Information, 1996)

Interpolating Spatial Data

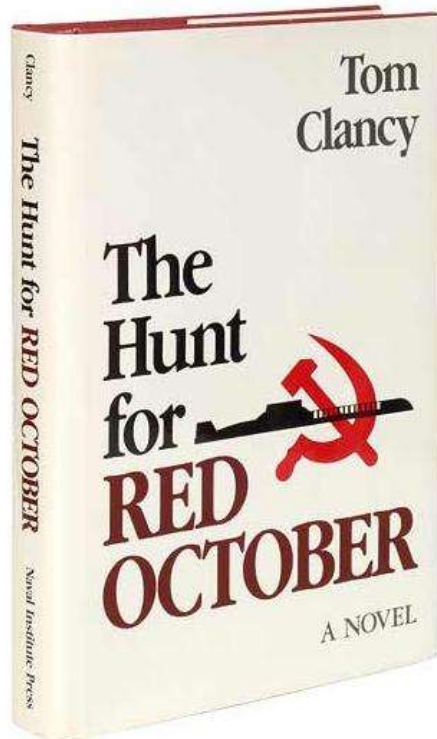
| Sample | Actual | Kriging | Error |
|--------|--------|---------|-------|
| #17 | 0 | 2 | (2) |
| 18 | 48 | 46 | (-2) |
| 19 | 64 | 65 | (1) |
| 20 | 65 | 56 | (-9) |
| 21 | 34 | 30 | (-4) |
| 22 | 0 | -1 | (-1) |
| 23 | 6 | 1 | (-5) |
| 24 | 79 | 70 | (-9) |
| 25 | 64 | 68 | (4) |
| 26 | 8 | 7 | (-1) |
| 27 | 19 | 19 | (0) |
| 28 | 6 | 3 | (-3) |
| 29 | 12 | 14 | (2) |
| 30 | 17 | 20 | (3) |
| 31 | 9 | 6 | (-3) |
| 32 | 14 | 11 | (-3) |



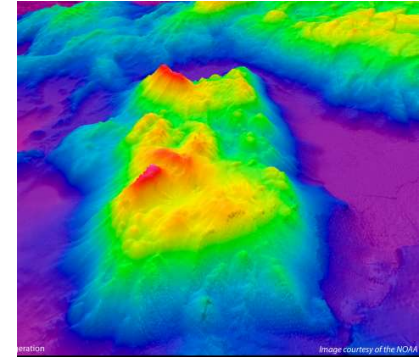
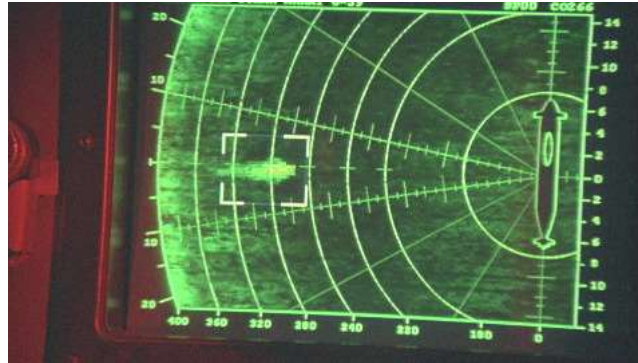
Interpolating Spatial Data



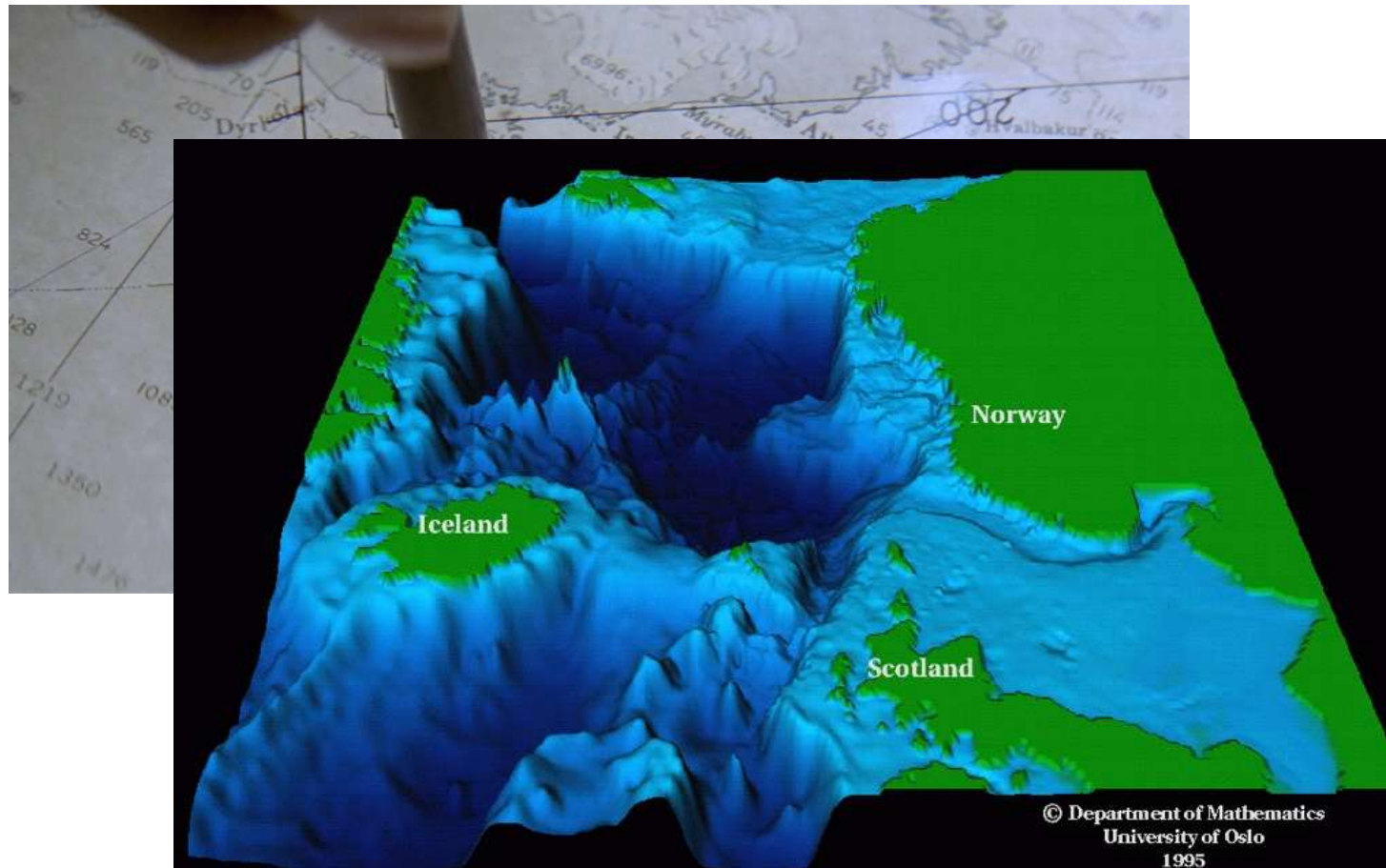
Interpolating Spatial Data



[Jonsey Reports](#)

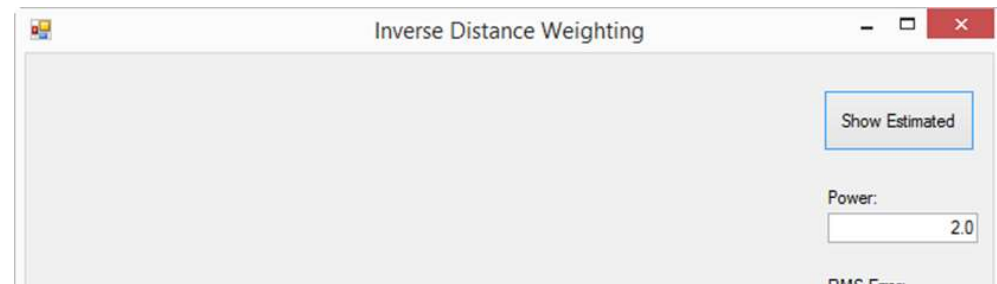


Interpolating Spatial Data



Lab - Inverse Distance Weighting

- Ocean area is **400** units square, partitioned into grid of **30 x 30** intervals



- Depth samples taken from random locations

- Floor reference height values

- Oblique

- PRNG seed

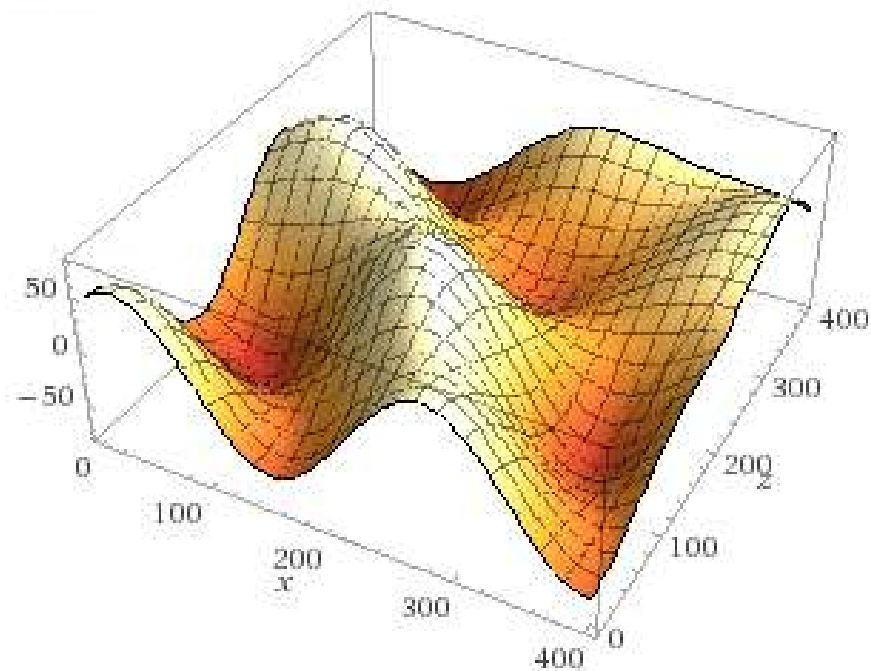
```
// Define the size of the ocean and the number of intervals to subdivide the ocean
double oceanSize = 400;
int intervals = 30;
double delta = oceanSize / intervals;

// Create an array of randomly sampled data points
int totalSamples = 220;
Point3D[] samples = new Point3D[totalSamples];
Random rnd = new Random(2015);
for (int i = 0; i < totalSamples; i++)
{
    double sampleX = rnd.NextDouble() * oceanSize;
    double sampleZ = -rnd.NextDouble() * oceanSize;
    double sampleY = GetActualHeight(sampleX, sampleZ);
    samples[i] = new Point3D(sampleX, sampleY, sampleZ);
}
```


The Interpolated Polynomial

$$y = 30 \sin\left(\frac{x}{4}\right) \cos\left(\frac{z}{4}\right) + 50 \cos\left(\frac{\sqrt{x^2 + z^2}}{4}\right)$$

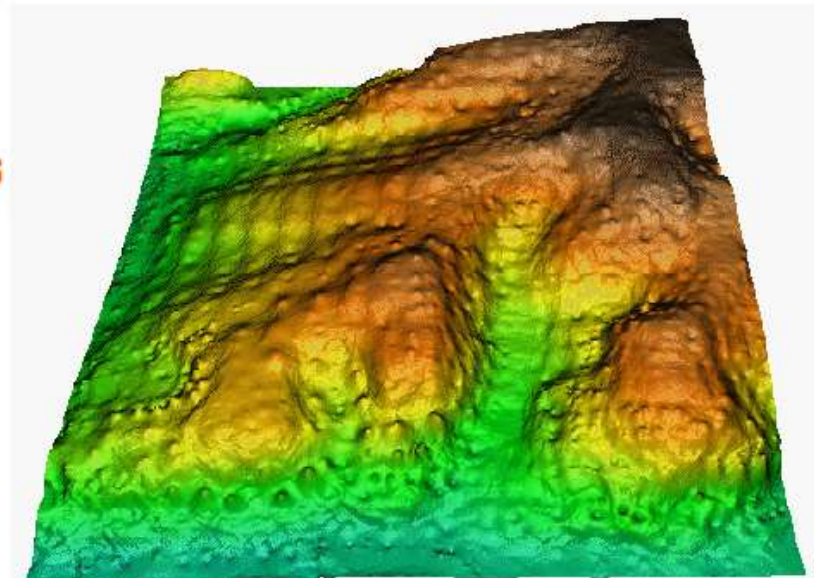
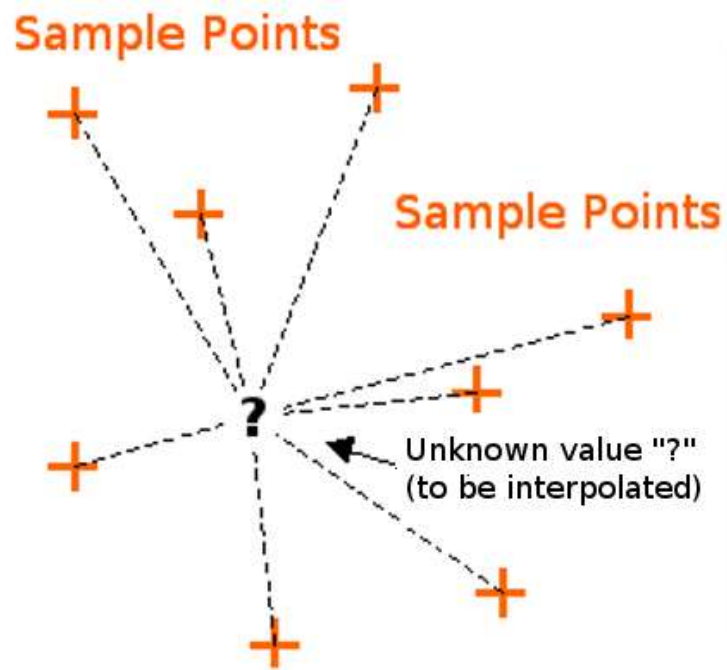
3D plot



The Inverse Distance Weighting (IDW) Method

- IDW is a type of deterministic method for **multivariate interpolation** with a known *scattered* set of points
- The assigned values to unknown points are calculated from a **weighted average** of the values available at the known points
- The theory is that the farther away a known point is from the unknown point, the **less** that distant known point can contribute to the unknown height
- *Closer* known points contribute *more* to the unknown height than known points farther away
- Your contribution **is inverse** to your distance

Interpolating Spatial Data



The Inverse Distance Weighting (IDW) Method

A general form of finding an interpolated value u at a given point x based on samples $u_i = u(x_i)$ for $i = 1, 2, \dots, N$ using IDW is an interpolating function:

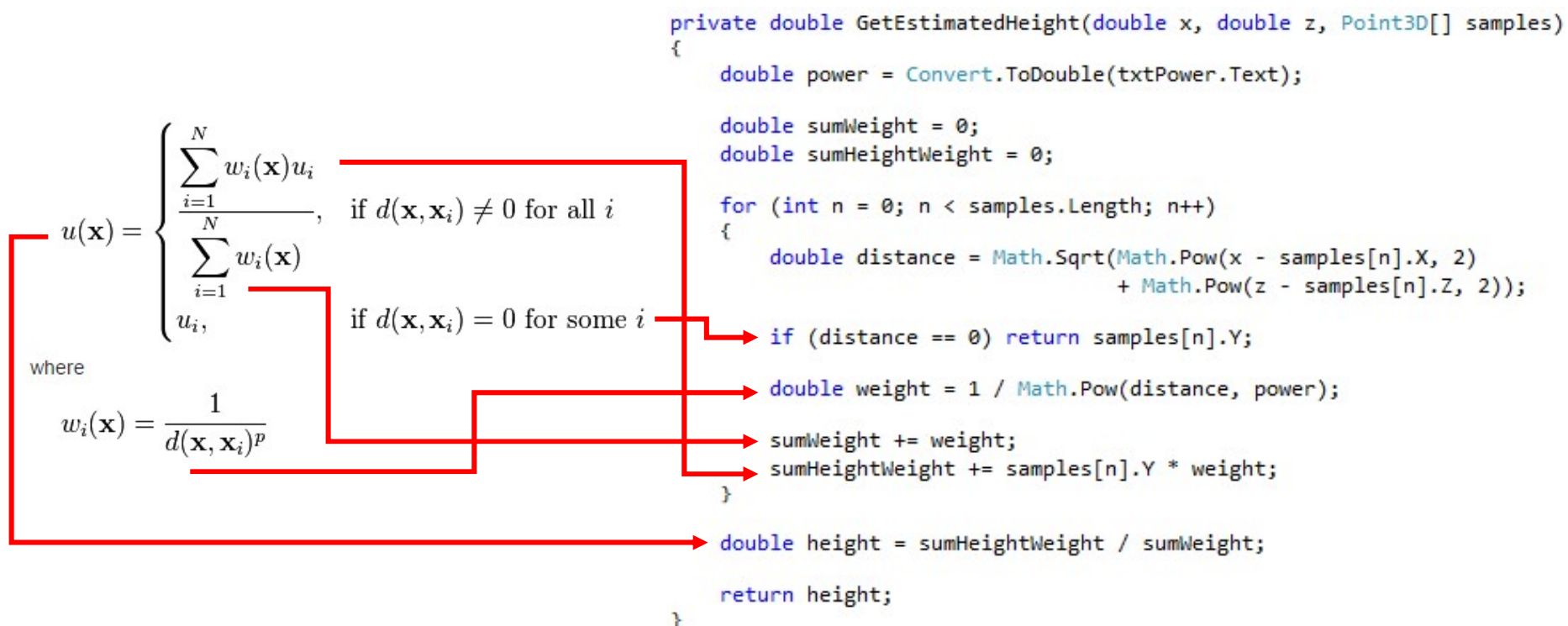
$$u(\mathbf{x}) = \begin{cases} \frac{\sum_{i=1}^N w_i(\mathbf{x}) u_i}{\sum_{i=1}^N w_i(\mathbf{x})}, & \text{if } d(\mathbf{x}, \mathbf{x}_i) \neq 0 \text{ for all } i \\ u_i, & \text{if } d(\mathbf{x}, \mathbf{x}_i) = 0 \text{ for some } i \end{cases}$$

where

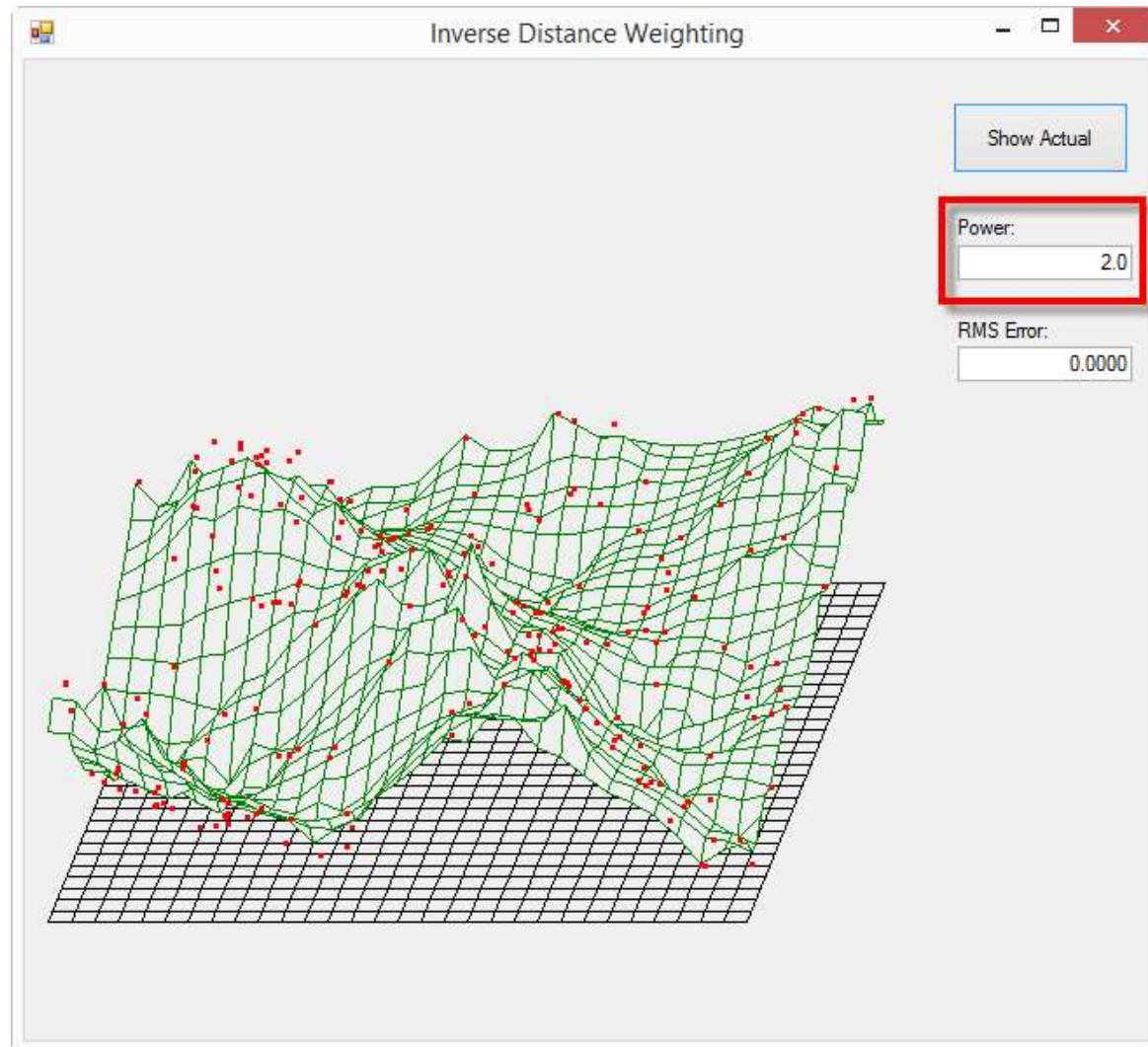
$$w_i(\mathbf{x}) = \frac{1}{d(\mathbf{x}, \mathbf{x}_i)^p}$$

**The p value
indicates the
"power" of
the distance
penalty**

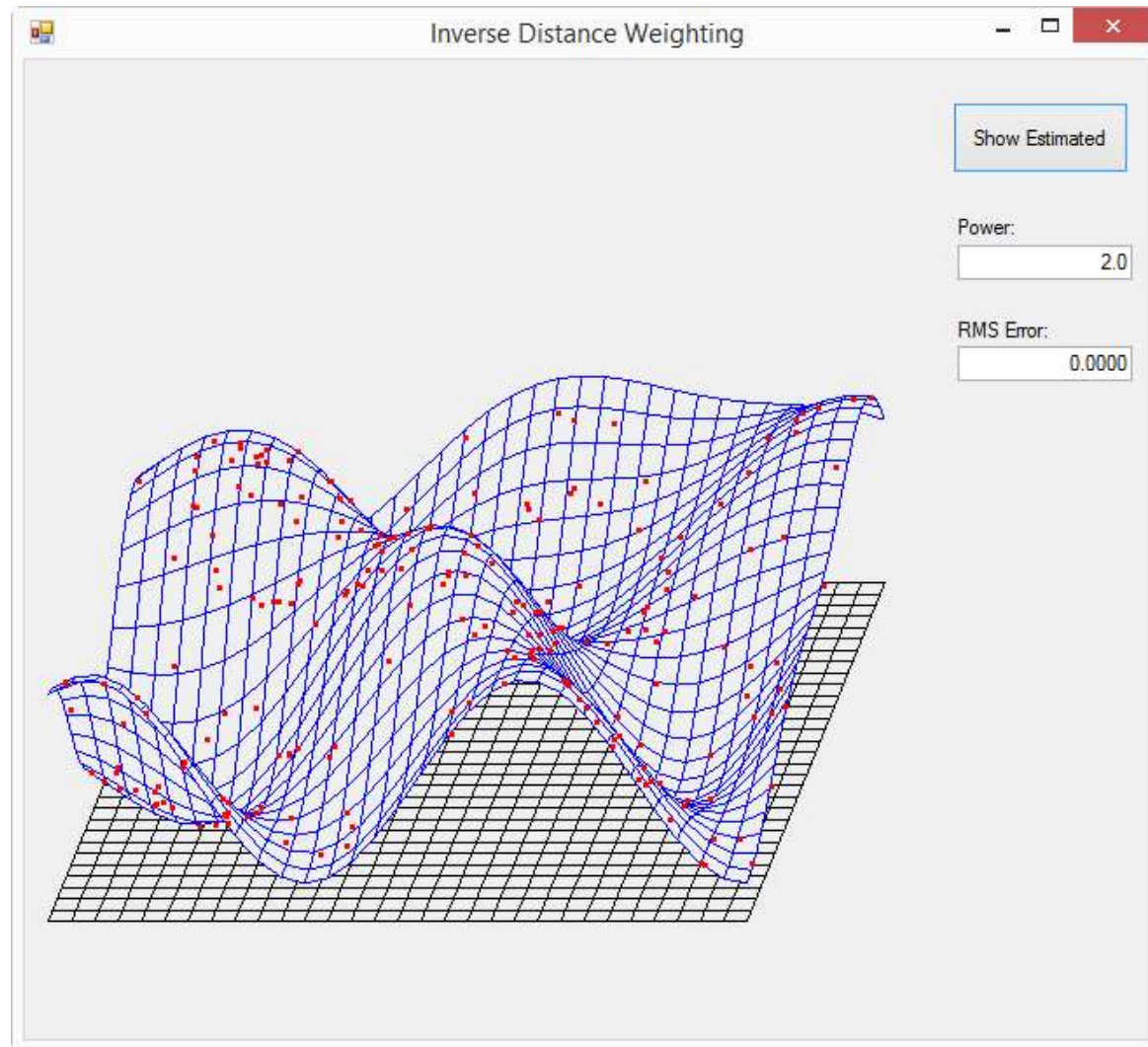
The Inverse Distance Weighting (IDW) Method



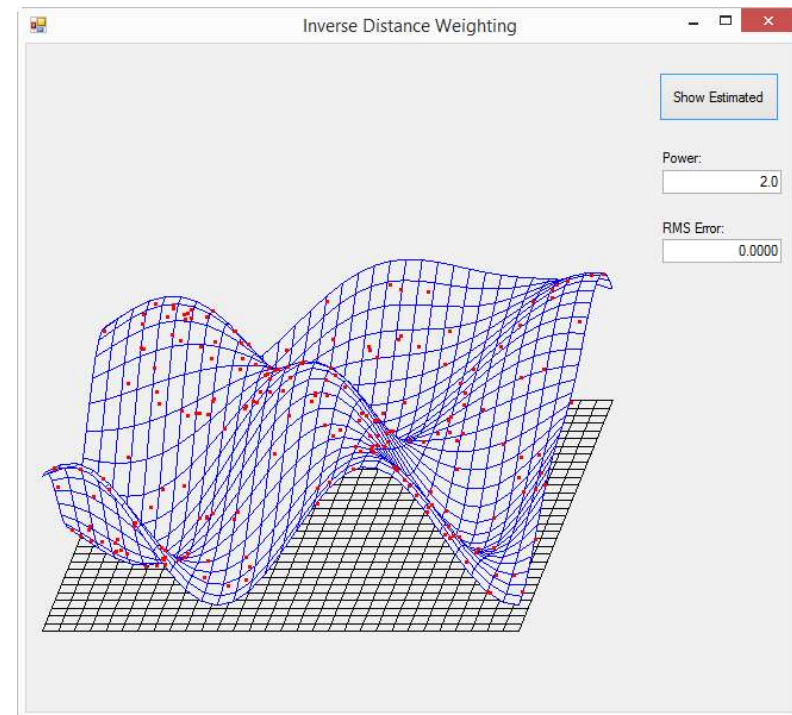
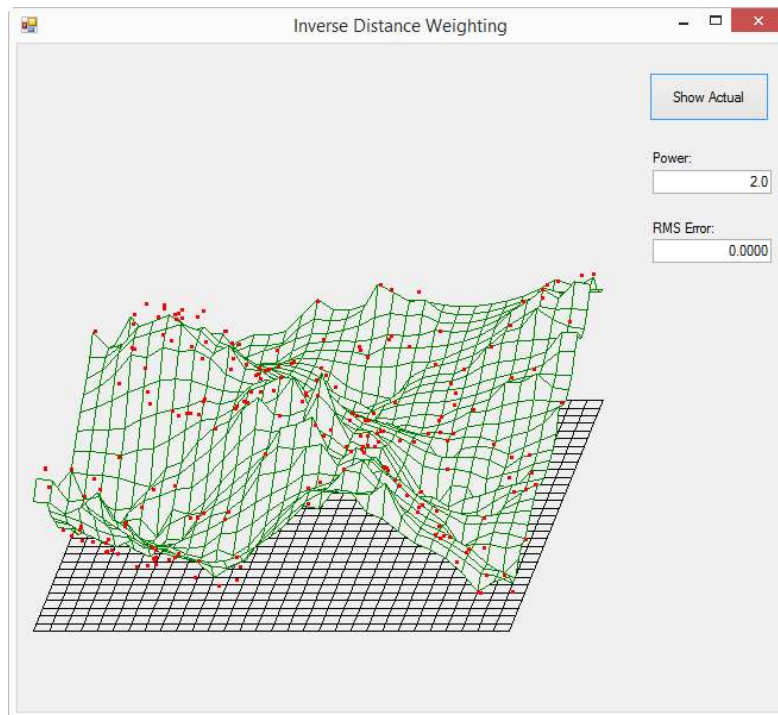
Initial Estimated Ocean Depth



“Actual” Ocean Depth

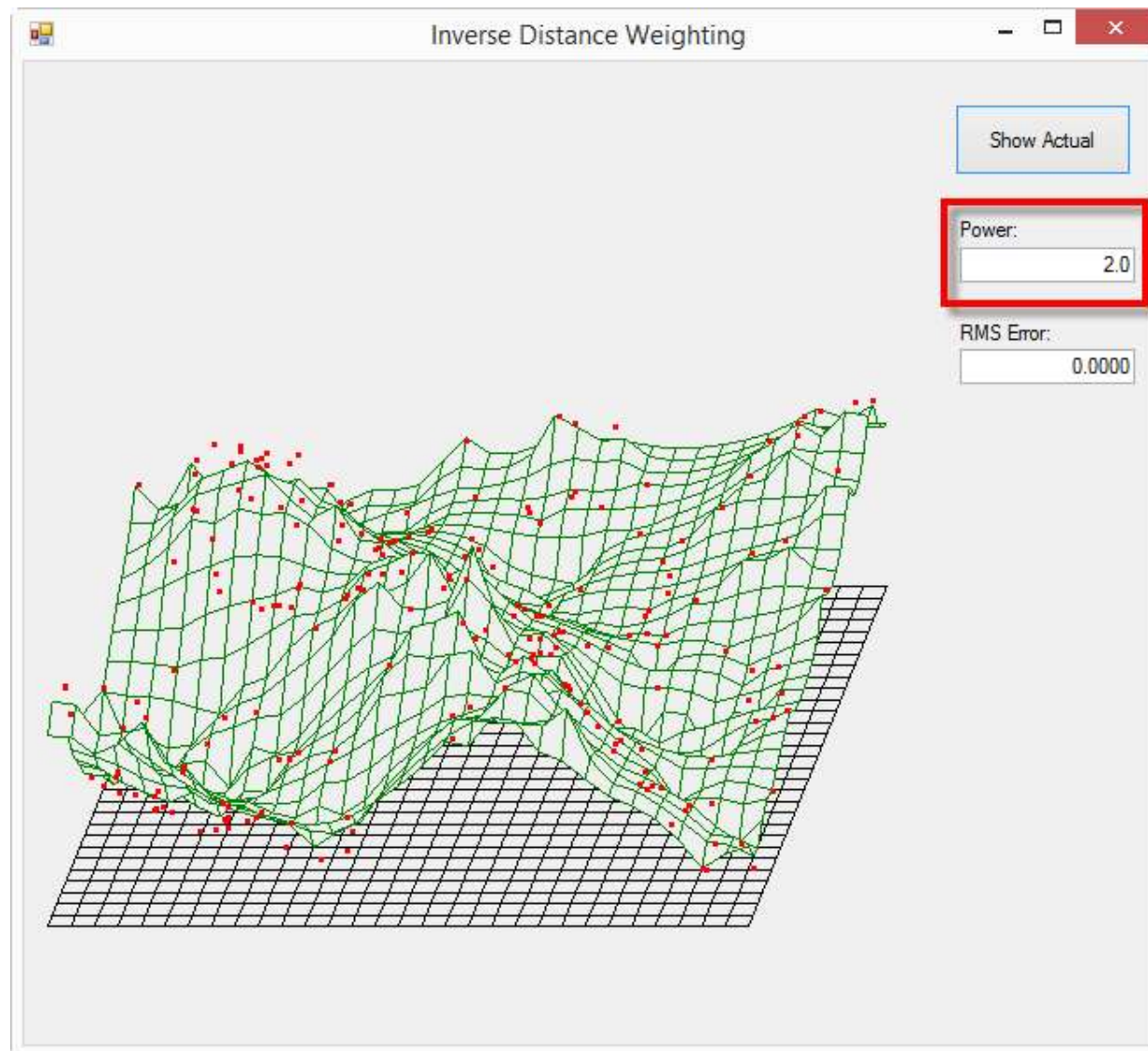


Estimate vs Actual (p = 2.0)

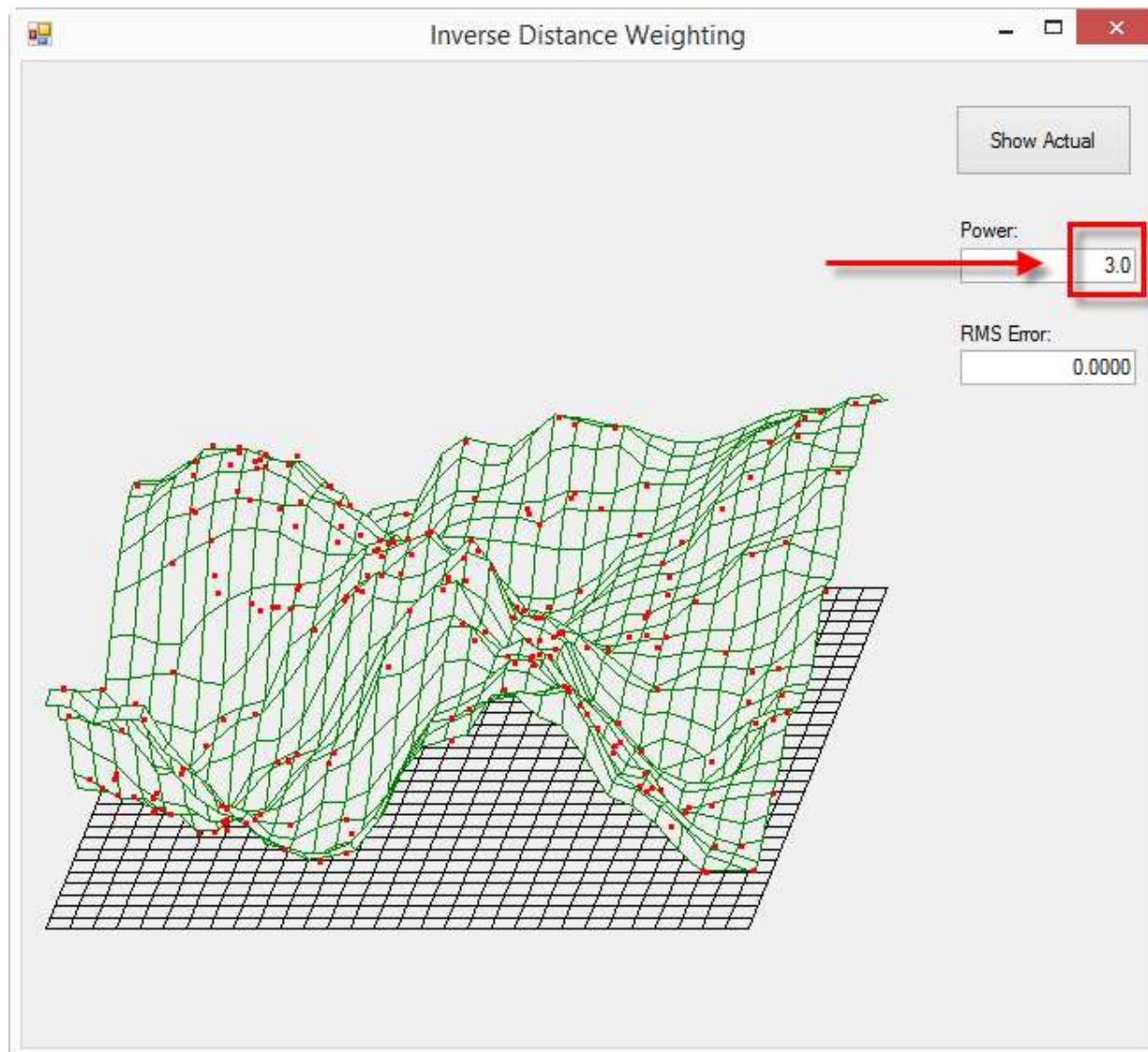


A first order approximation having only 24% of the world sampled (220 of 900 actual points)

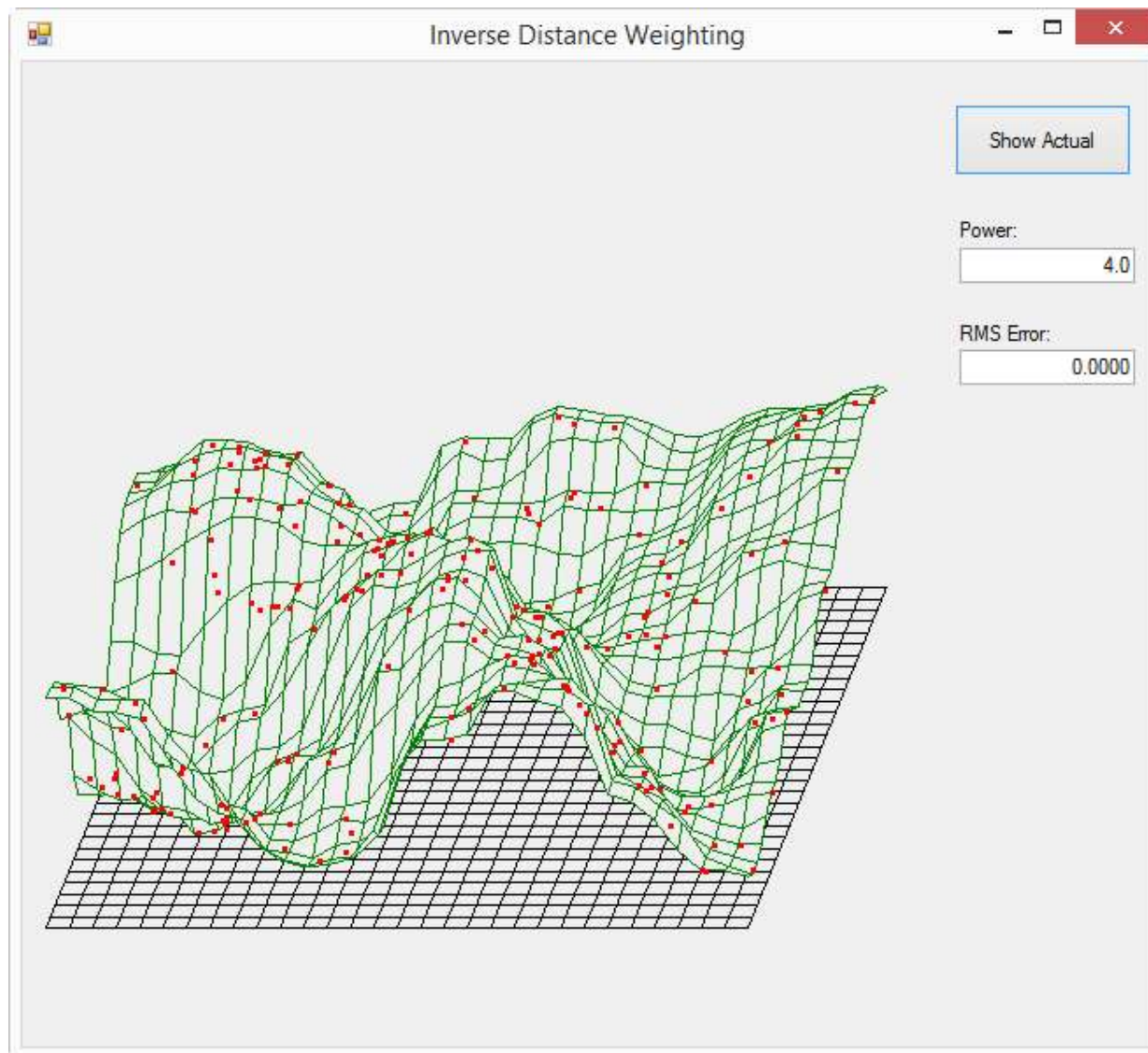
Estimate ($p = 2.0$)



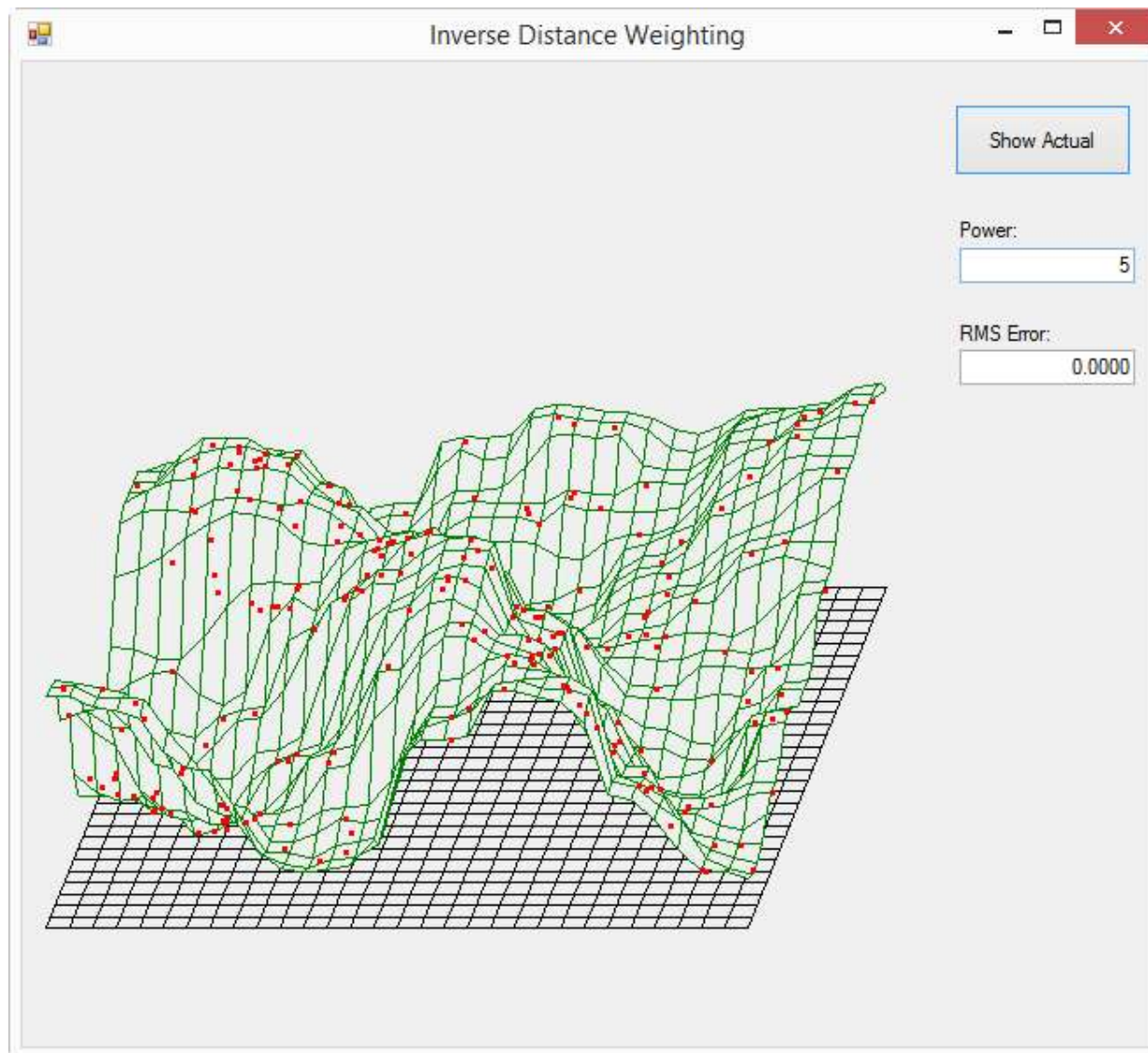
Estimate ($p = 3.0$)



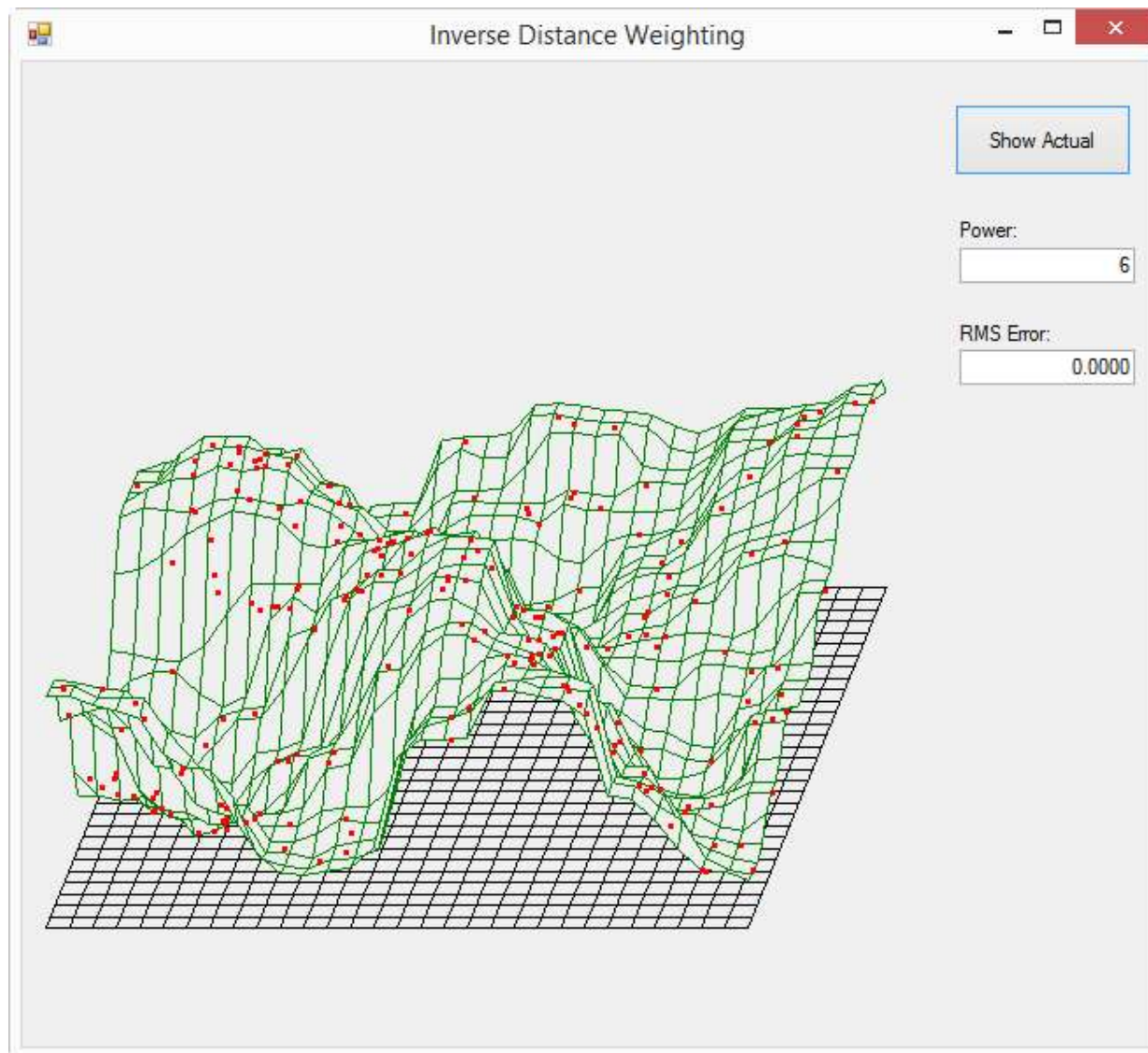
Estimate ($p = 4.0$)



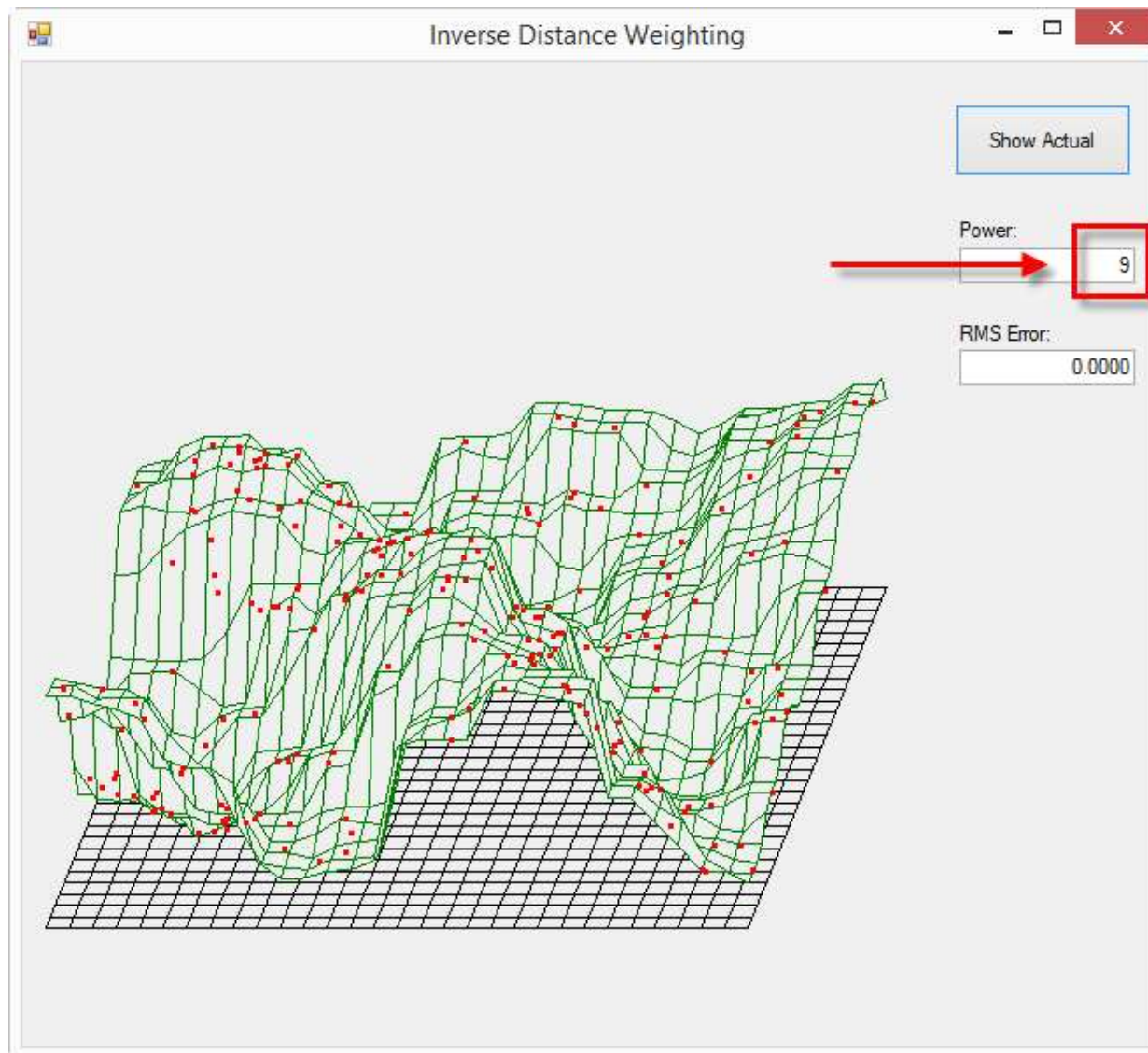
Estimate ($p = 5.0$)



Estimate ($p = 6.0$)



Estimate ($p = 9.0$)



Root Mean Square Deviation

- As we increase the *power* term **p**, is our model getting better or worse at predicting reality?
- The root-mean-square deviation (**RMSD**) is a statistic to measure the differences between values **predicted** by a model and the values actually **observed**

$$\text{RMSD} = \sqrt{\frac{\sum_{t=1}^n (\hat{y}_t - y)^2}{n}}.$$

- If we sum $(\text{actual} - \text{estimated})^2$ for all sample points, we can calculate a comparative statistic to empirically determine the optimal **p** value that minimizes the overall error of the model

Lab – Inverse Distance Weighting

- Write the code to calculate the **RMSD**
- The function is provided the 3D arrays containing the actual and estimated ocean depth soundings

```
private double CalcRMSD(Point3D[,] actual, Point3D[,] estimated)
{
    // TODO: You must complete the function to calculate the RMS deviation
    double rmsError = 0;
    return rmsError;
}
```

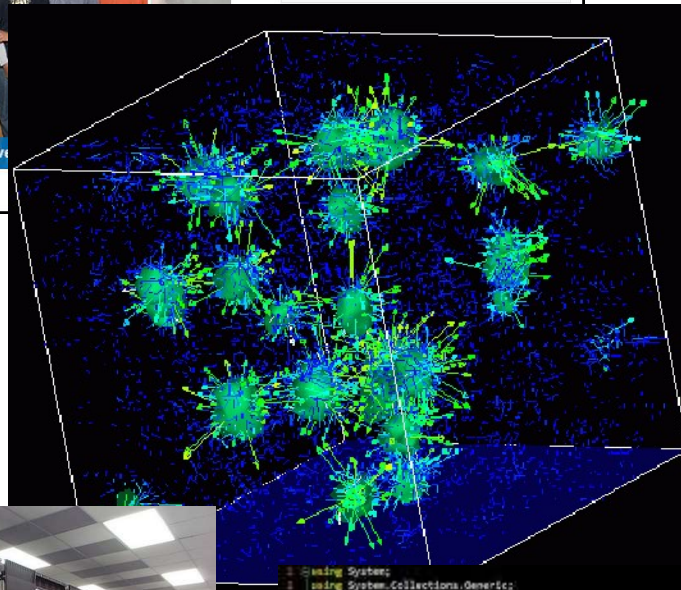
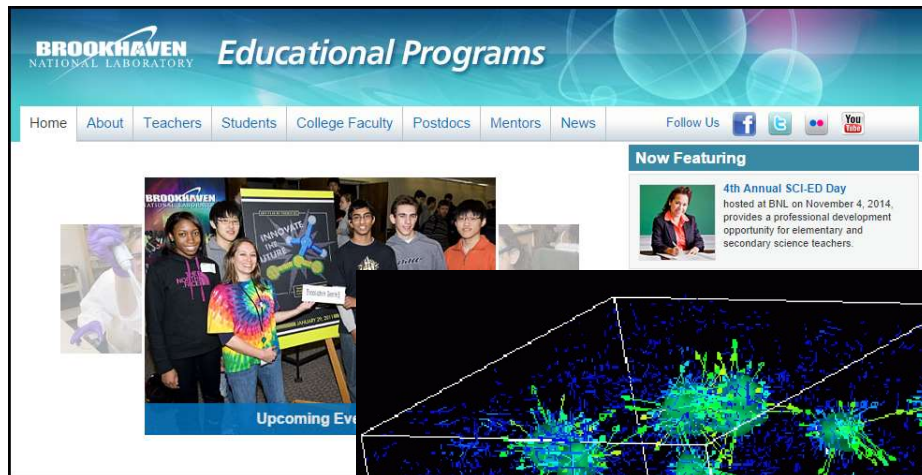
- Once the application is displaying the **RMSD**, for the current sample size of **220** and the actual ocean floor approximated with **30 x 30** intervals, what value of **p** minimizes the **RMSD**?

Lab – Additional Research Questions

- What happens to the “fit” of the model if **p** is too high? Why does this occur?
- With **p** constant, what happens to the **RMSE** as you increase or decrease the number of **samples**?
- With **p** constant, what happens to the **RMSE** as you increase or decrease the number of **intervals**?
- With **p = 2**, what **ratio** of sample count divided by interval count minimizes the **RMSE**?

So now you know

- One **method** for interpolating multi-dimensional data taken from random sample locations
- The mathematics of the Inverse Distance Weighting (**IDW**) method
- How to convert non-uniformly measured spatial data to a **regular conforming mesh**
- How to use **RMSD** as one metric to characterize the “goodness of fit” for predicted interpolated data points
- Just like **Red October** – scientists rarely enjoy the luxury of having too many data samples – we must often interpolate to fill in the missing gaps



```

1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Windows.Forms;
9
10 namespace SimpleEvents
11 {
12     public partial class Form1 : Form
13     {
14         Person person = new Person();
15
16         public Form1()
17         {
18             InitializeComponent();
19             person.FirstName = "Christian";
20             person.LastName = "Foss";
21         }
22
23         private void button1_Click(object sender, EventArgs e)
24         {
25             person.HairColor = textBox1.Text;
26         }
27     }
28 }

```

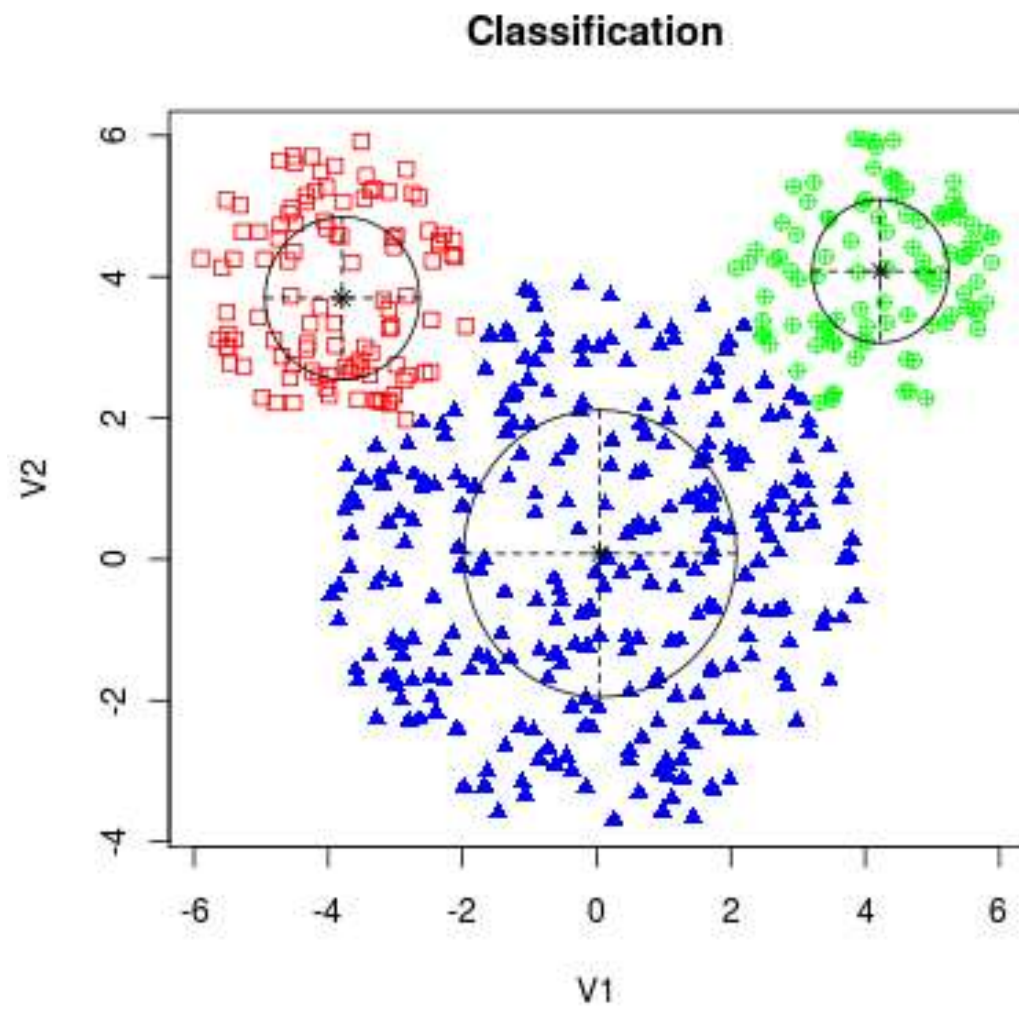
Introducing Scientific Computing

Part 4 k-Means Clustering

k-means Clustering

- k -means clustering aims to [partition](#) n observations into k clusters in which each observation belongs to the cluster with the nearest [mean](#)
- The problem is computationally difficult ([NP-hard](#)); however, there are efficient [heuristic algorithms](#) that are commonly employed and converge quickly to a [local optimum](#)
- The term " k -means" was first used by James MacQueen in 1967, though the idea goes back to [Hugo Steinhaus](#) in 1957
- A more efficient version was published Hartigan and Wong in 1979

k-means Clustering



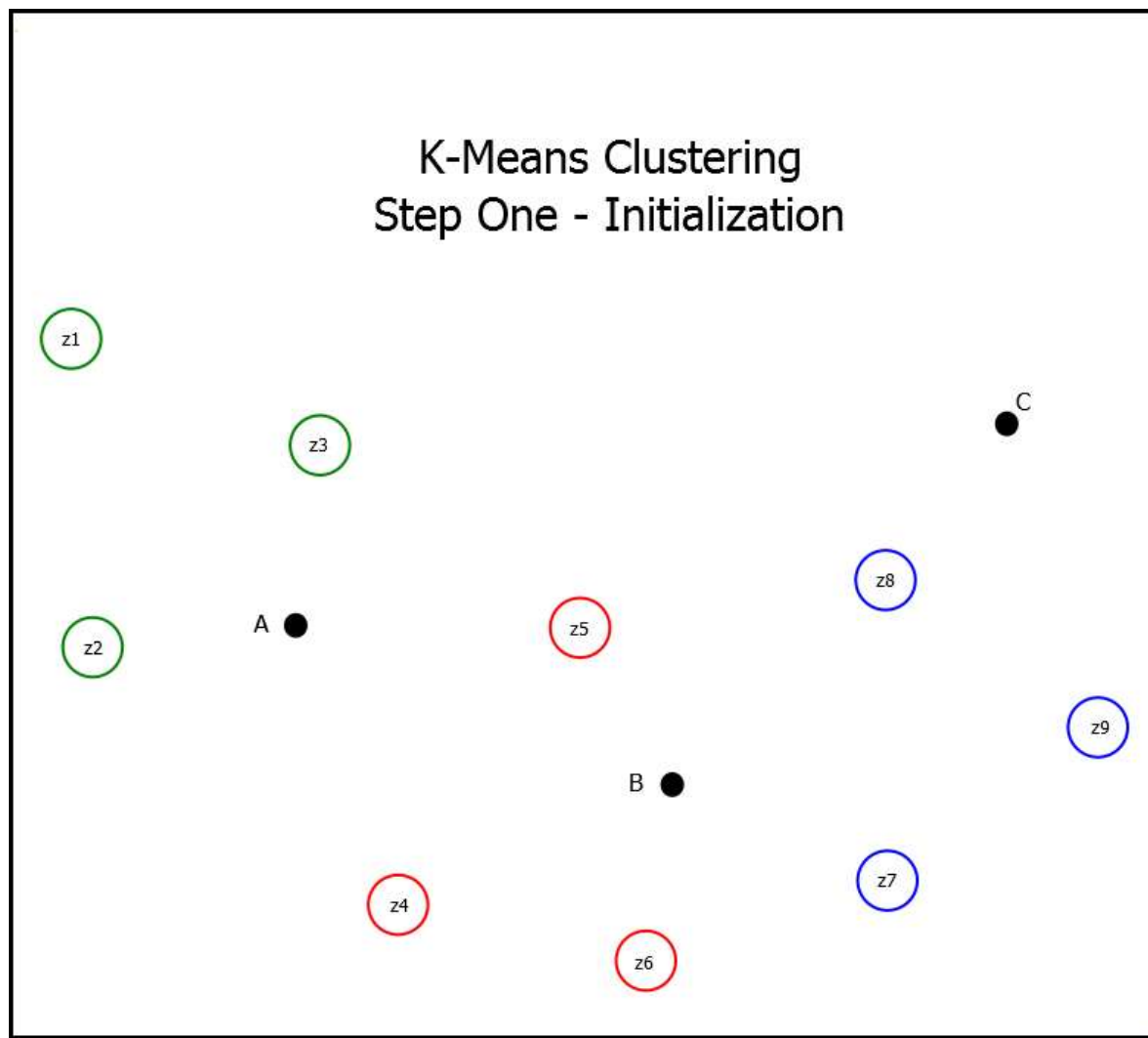
k-means Clustering

- The basic approach is to associate a “cluster #” with each data point
 - The cluster # is between 0 and $k - 1$ inclusively
 - If we wanted to cluster our data into 4 regions, each data point would get assigned a cluster # between 0 and 3
 - The cluster # can be assigned initially at random to every data point, **but** every cluster # must have **at least one** data point assigned to it
- The initial means of *each* cluster are calculated by taking the average values (in each of the dimensions) of the sample data points that assigned to that cluster

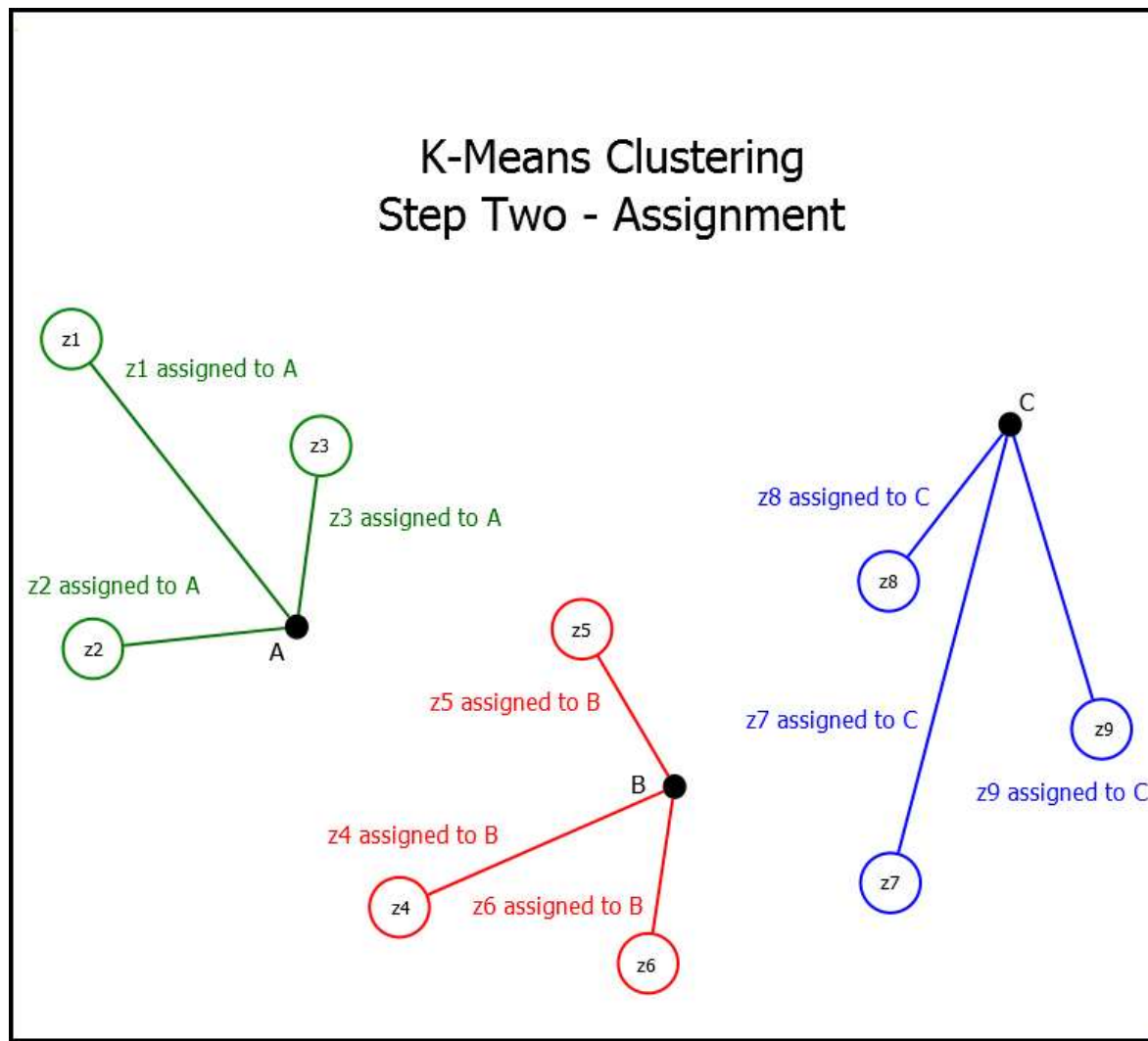
k-means Clustering

- After this initial setup, the algorithm begins to iterate
 - Every data point is reassigned to the cluster # which has the *means* **closest** (in distance) to that data point
 - After all data points are assigned (possibly new) cluster #s, the means for each cluster are **recalculated**
 - The algorithm then loops again
- Eventually the algorithm will reach a state where no data point has its cluster # changed
 - Once this happens, we have obtained steady-state **convergence**
 - Every data point will now belong to one and only one k-cluster
- We can color each data point by the cluster to which it most closely associated per the means

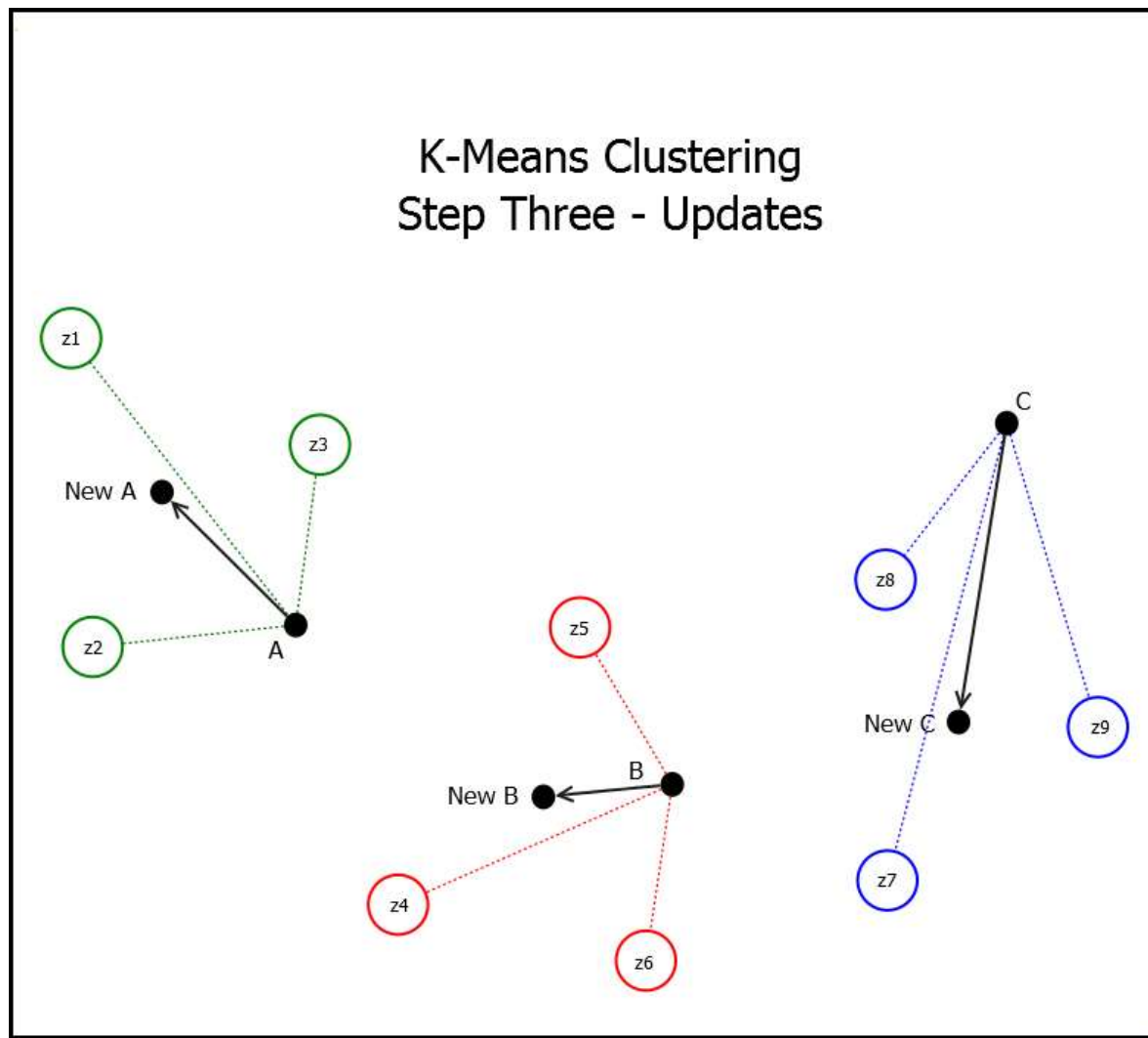
k-means Clustering



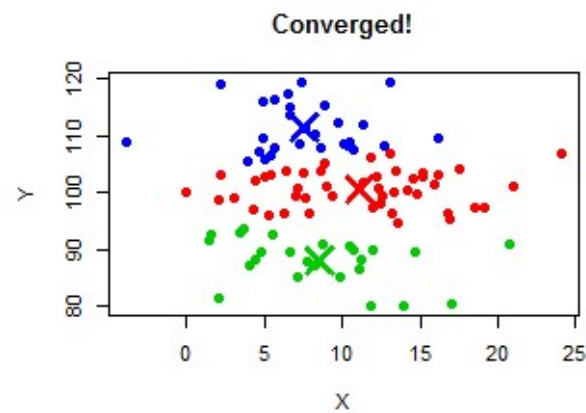
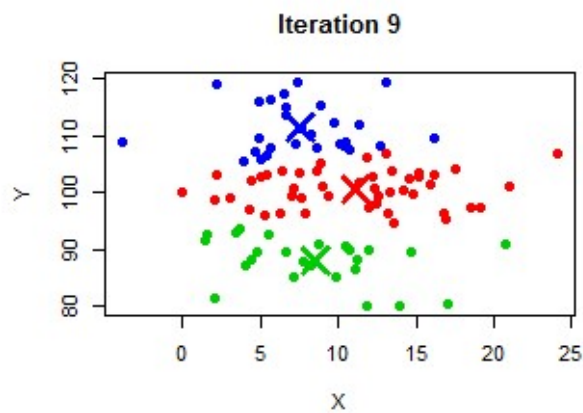
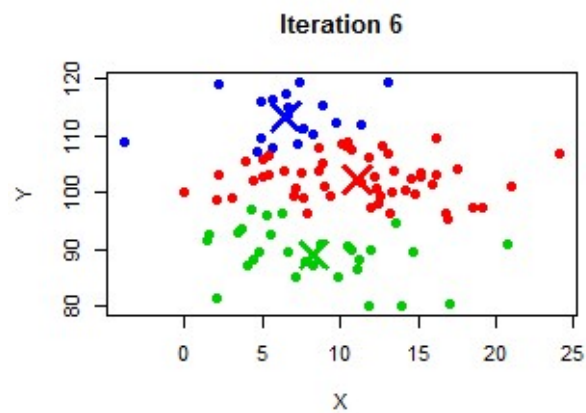
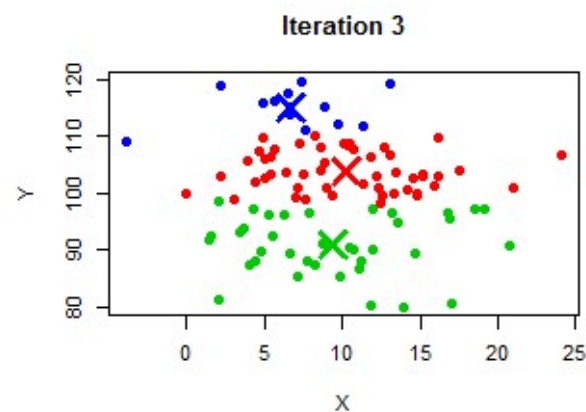
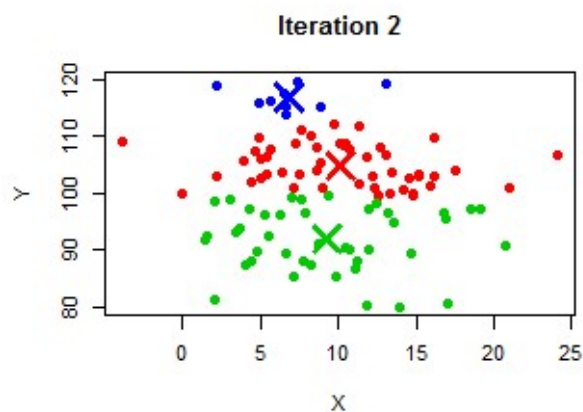
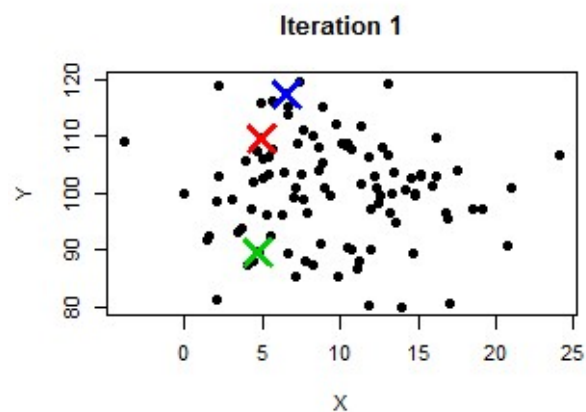
k-means Clustering



k-means Clustering



k-means Clustering



k-means Clustering

```
private void Form1_Load(object sender, EventArgs e)
{
    rawData[0] = new double[] { 65.0, 220.0 };
    rawData[1] = new double[] { 73.0, 160.0 };
    rawData[2] = new double[] { 59.0, 110.0 };
    rawData[3] = new double[] { 61.0, 120.0 };
    rawData[4] = new double[] { 75.0, 150.0 };
    rawData[5] = new double[] { 67.0, 240.0 };
    rawData[6] = new double[] { 68.0, 230.0 };
    rawData[7] = new double[] { 70.0, 220.0 };
    rawData[8] = new double[] { 62.0, 130.0 };
    rawData[9] = new double[] { 66.0, 210.0 };
    rawData[10] = new double[] { 77.0, 190.0 };
    rawData[11] = new double[] { 75.0, 180.0 };
    rawData[12] = new double[] { 74.0, 170.0 };
    rawData[13] = new double[] { 70.0, 210.0 };
    rawData[14] = new double[] { 61.0, 110.0 };
    rawData[15] = new double[] { 58.0, 100.0 };
    rawData[16] = new double[] { 66.0, 230.0 };
    rawData[17] = new double[] { 59.0, 120.0 };
    rawData[18] = new double[] { 68.0, 210.0 };
    rawData[19] = new double[] { 61.0, 130.0 };

    rawData[20] = new double[] { 56.0, 247.0 }; // An outlier!

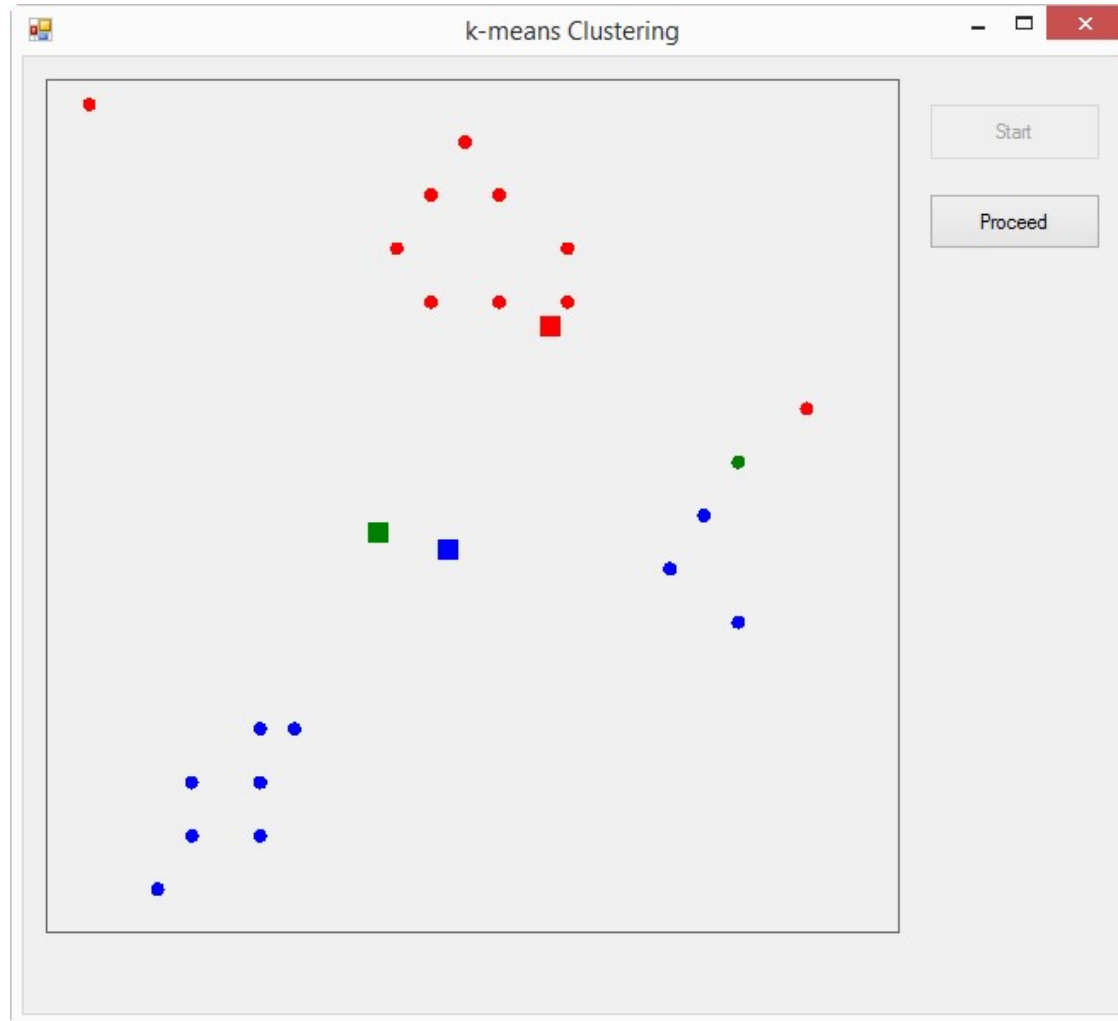
    DrawData();

    buttonProceed.Enabled = false;
}
```


k-means Clustering – Raw Data



k-means Clustering – Iteration 1



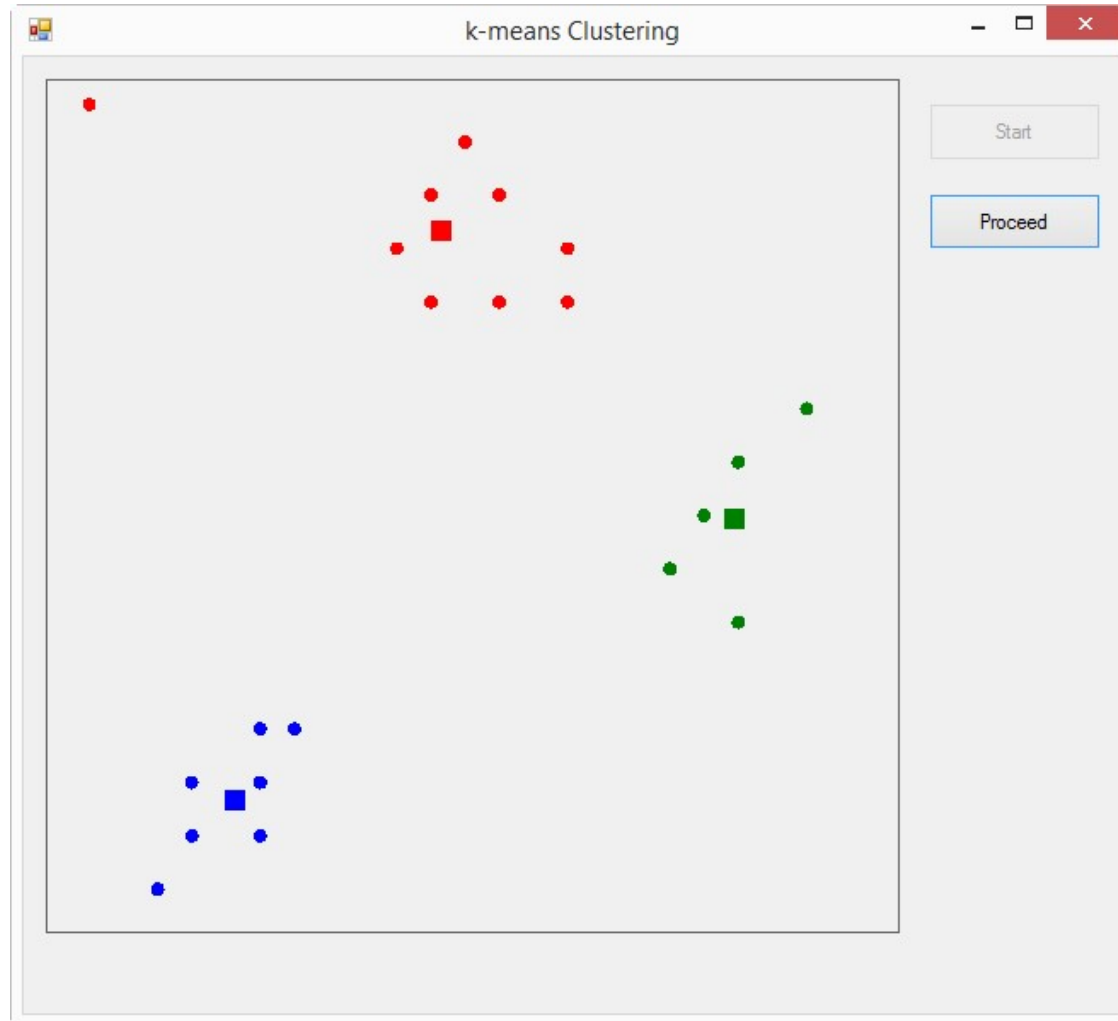
k-means Clustering – Iteration 2



k-means Clustering – Iteration 3



k-means Clustering – Iteration 4



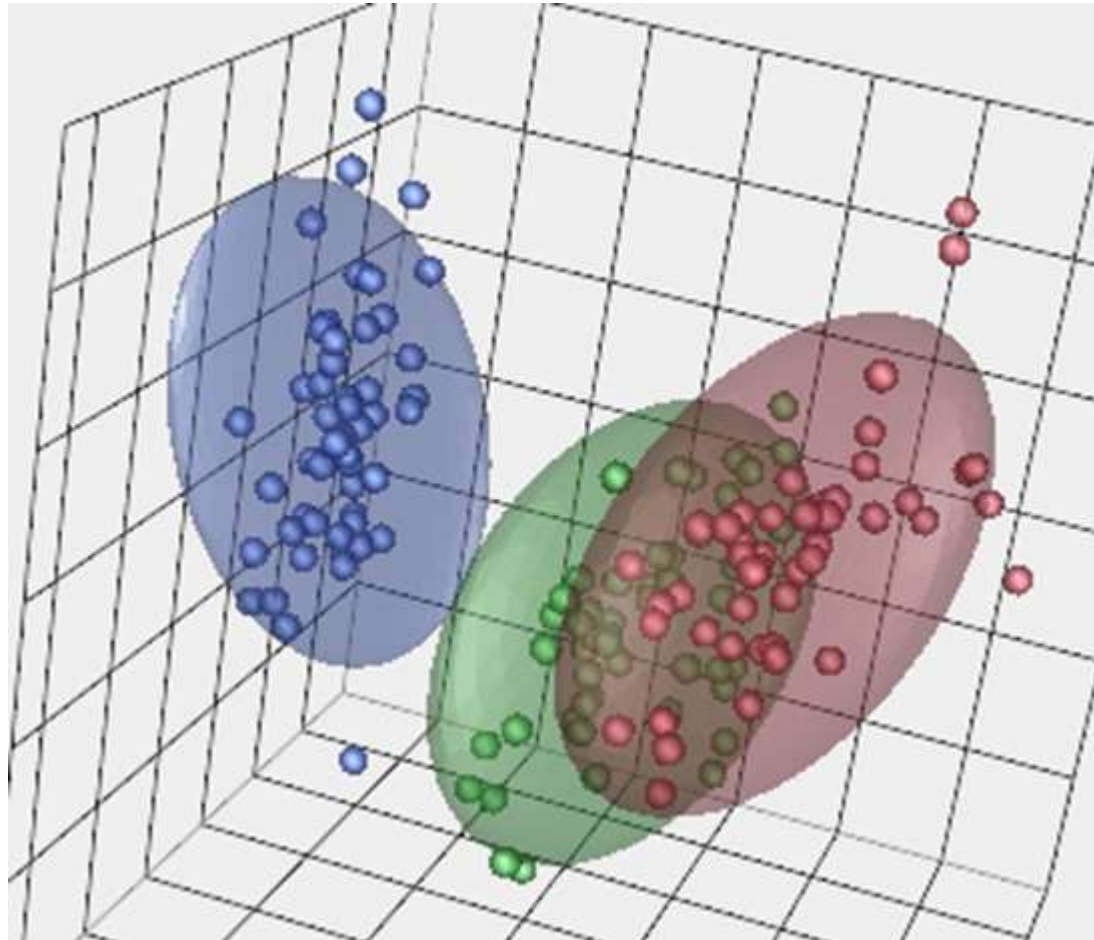
k-means Clustering - Converged



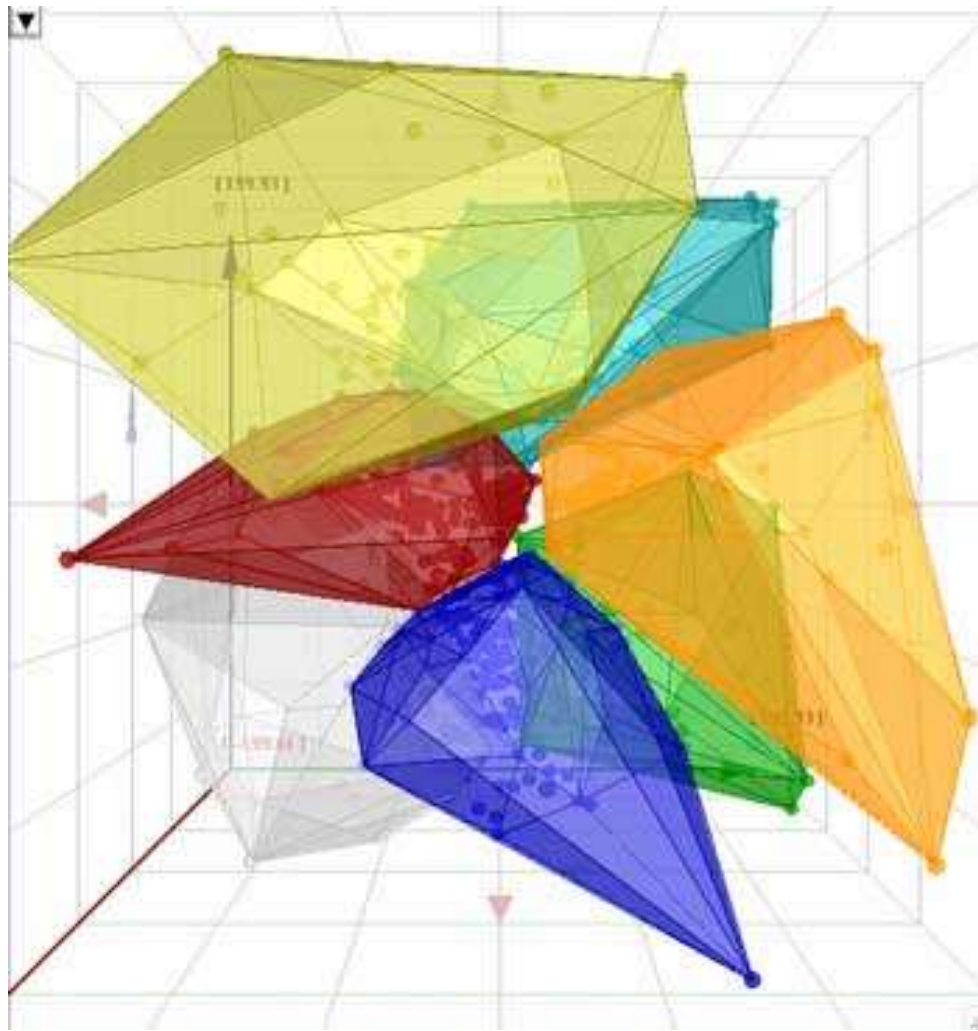
Identifying Data Outliers

- Once we have data partitioned into clusters, we can eliminate outliers by evicting any data points whose distance from the cluster's mean is greater than a given standard deviation
 - The data points should be distributed around the cluster's mean in a normal distribution
 - 99.97% of data points should be within 3 sigmas from the mean
 - If a data point is > 3 sigmas from the mean of its cluster, it probably is an outlier and should be evicted and a new cluster # should be assigned to it
- If a data point winds up getting evicted from all clusters, it is probably a data capture error and could be a statistical anomaly

3-Dimensional k-means Clustering

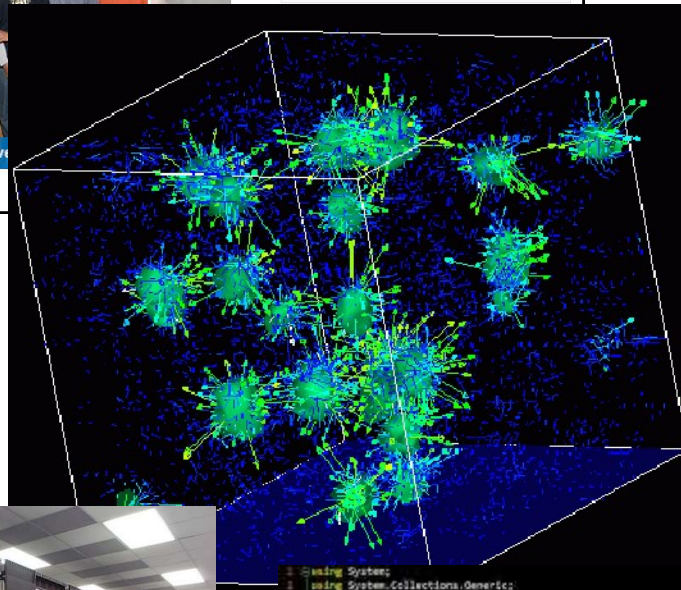
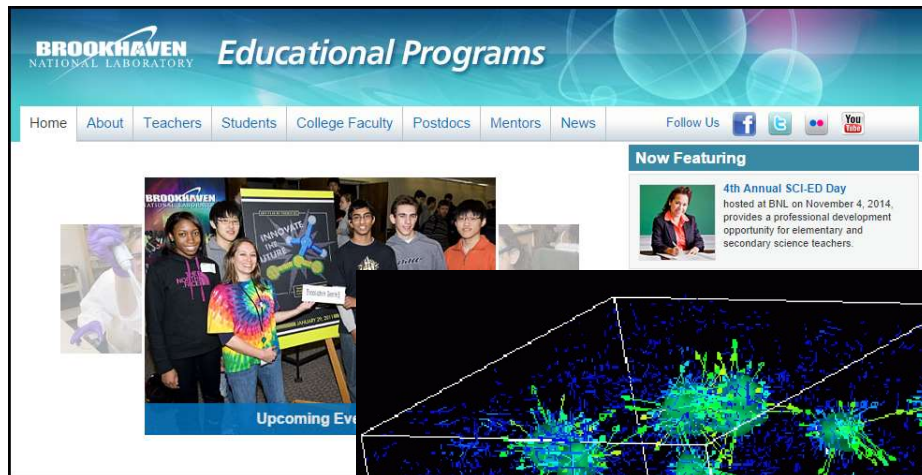


n-Dimensional k-means Clustering



Lab - k-means Clustering

- Open “**kMeansClustering.sln**” and view the code-behind file “**Form1.cs**”
- Set **numClusters** = 1 – what happens?
- Increase numClusters from 2 to 5 in unit steps
 - What happens at five clusters?
 - Why do you think this happens?
- What statistic could we use to determine the “optimal” number of k-Means clusters for a given dataset?



```

1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Windows.Forms;
9
10 namespace SimpleEvents
11 {
12     public partial class Form1 : Form
13     {
14         Person person = new Person();
15
16         public Form1()
17         {
18             InitializeComponent();
19             person.FirstName = "Christian";
20             person.LastName = "Foss";
21         }
22
23         private void button1_Click(object sender, EventArgs e)
24         {
25             person.HairColor = textBox1.Text;
26         }
27     }
28 }

```

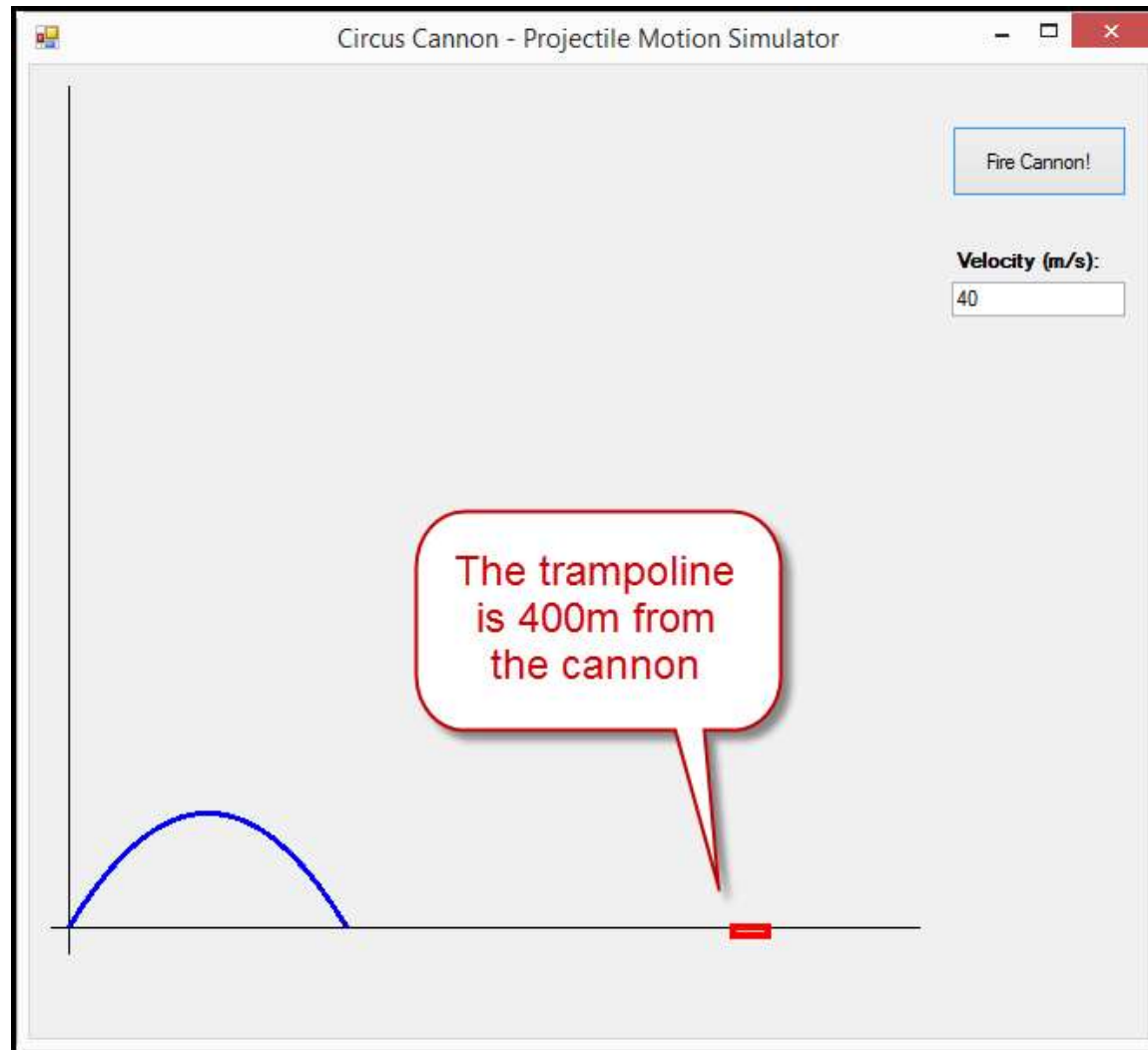
Introducing Scientific Computing

Part 5 Circus Cannon

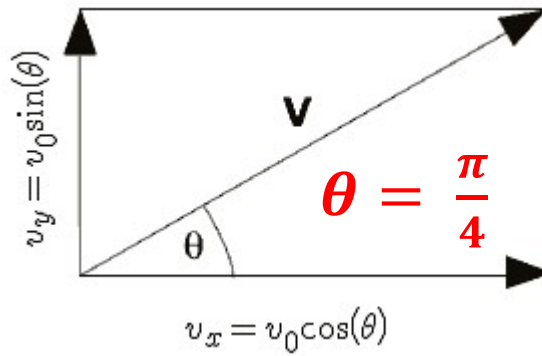
Lab – Circus Cannon



Lab – Circus Cannon



Lab – Circus Cannon



$$h = \frac{v_0^2 \sin^2(\theta)}{2g}$$

$$\text{Range} = \frac{4h}{\tan(\theta)}$$

Given Range = 400m,
what does v_0 need to be?

$$x = v_0 \cdot t \cdot \cos(\theta)$$

$$y = v_0 \cdot t \cdot \sin(\theta) - \frac{1}{2}gt^2$$

$$t = \frac{x}{v_0 \cdot \cos(\theta)}$$

$$y = x \cdot \tan(\theta) - \frac{g}{2 \cdot v_0^2 \cdot \cos^2(\theta)}$$

This is the equation of motion that allows us to plot y as x increases from launch point to trampoline

Lab – Circus Cannon

```
// Get velocity from text box
double initialVelocity = Convert.ToSingle(textBox1.Text);

// Set fixed angle of elevation (45 degrees converted to radians)
double theta = 45.0 * Math.PI / 180.0;

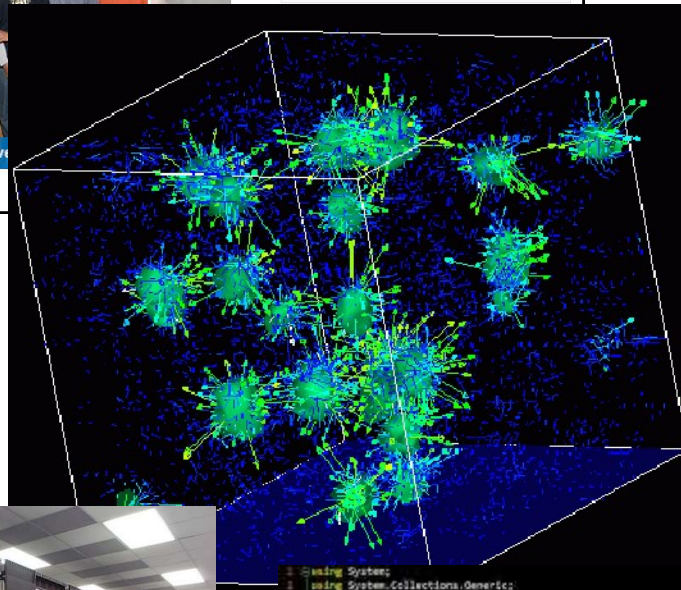
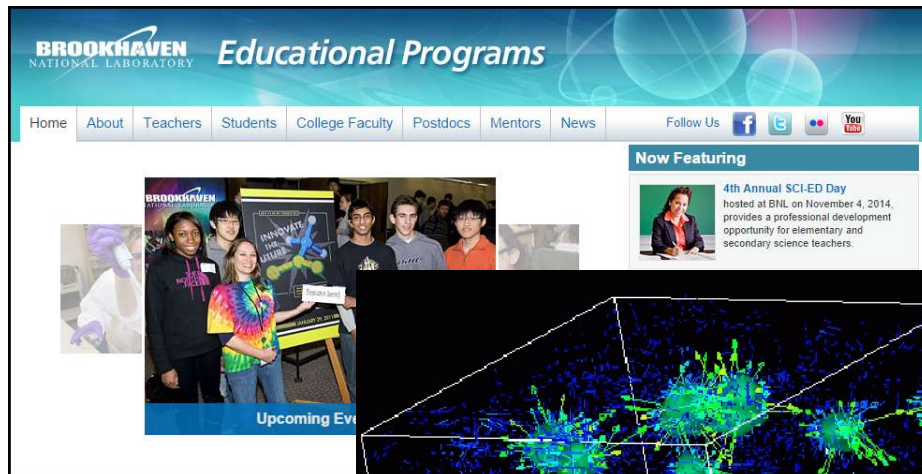
// Set acceleration due to gravity in SI units
double gravity = 9.81;

// Calculate height and range of trajectory
double trajectoryHeight = Math.Pow(initialVelocity, 2) * Math.Pow(Math.Sin(theta), 2) / (2 * gravity);
double trajectoryRange = 4 * trajectoryHeight / Math.Tan(theta);
```

```
// Graph the trajectory of the man shot from the circus cannon
for (int i = 0; i <= intervals; i++)
{
    // Calculate WORLD coordinates for current x and f(x)
    double worldX = worldDeltaX * i;
    double worldY = Math.Tan(theta) * worldX -
        (gravity / (2 * Math.Pow(initialVelocity, 2) * Math.Pow(Math.Cos(theta), 2))) * Math.Pow(worldX, 2);

    // Convert WORLD coordinates to SCREEN coordinates
    double screenX = (worldX - worldXmin) * scaleX;
    double screenY = pictureBox1.Height - (worldY - worldYmin) * scaleY;

    // Add this SCREEN coordinate to array of points
    screenPoints[i] = new PointF((float)screenX, (float)screenY);
}
```

```

1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Windows.Forms;
9
10 namespace SimpleEvents
11 {
12     public partial class Form1 : Form
13     {
14         Person person = new Person();
15
16         public Form1()
17         {
18             InitializeComponent();
19             person.FirstName = "Christian";
20             person.LastName = "Foss";
21         }
22
23         private void button1_Click(object sender, EventArgs e)
24         {
25             person.HairColor = textBox1.Text;
26         }
27     }
28 }

```

Introducing Scientific Computing

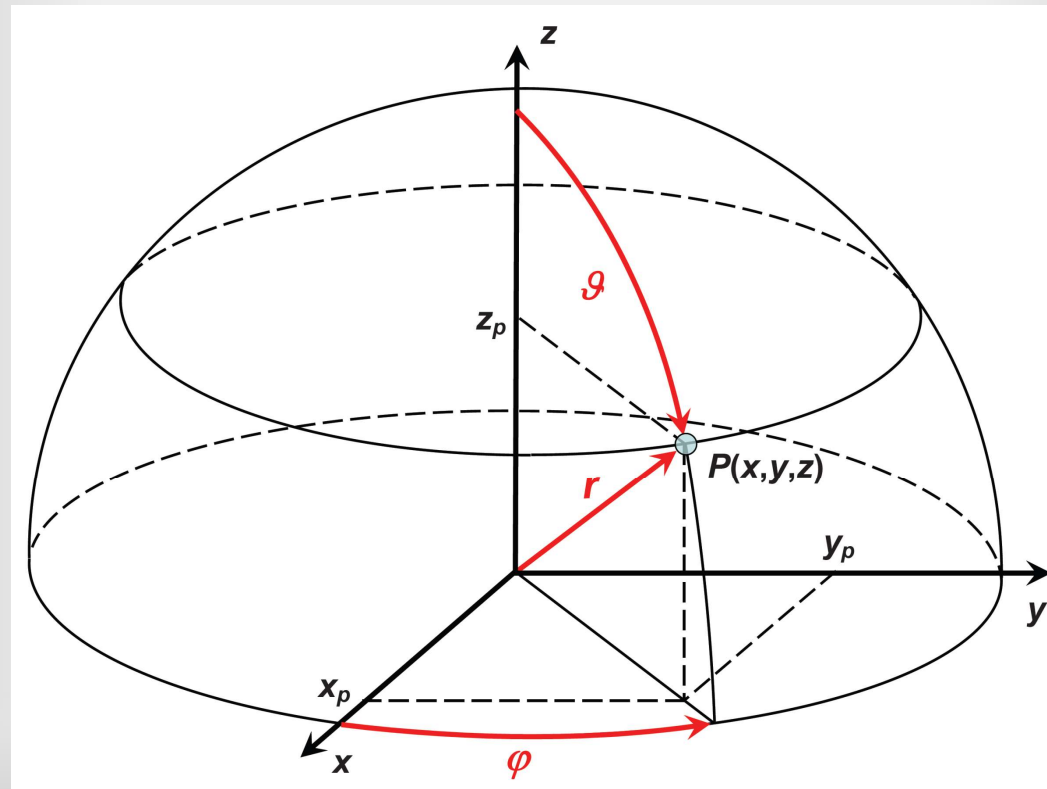
Part 6 Draw Your Own Death Star

Wireframe Graphics, Backface Culling, and Facet Shading

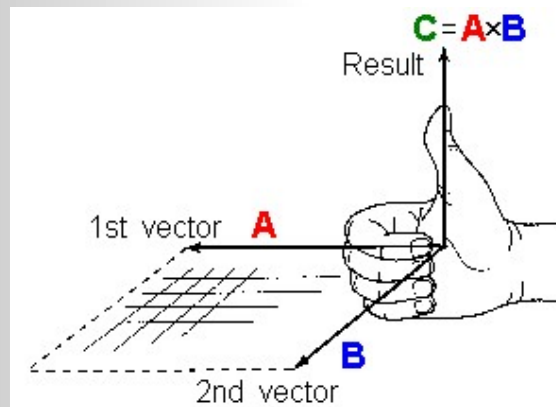
Class Goals

- Understand x, y, z axis orientation
- Explain oblique projection (2.5D)
- Relate vertexes to faces when drawing a monolith
- Use spherical coordinates to draw wireframes
- Correlate intervals with vertices and facets
- Identify a surface normal using the right-hand rule
- Relate dot product between vectors to orientation angle
- Establish a camera vector
- Calculate cross product and dot product
- Perform back face culling and facet shading

Spherical Coordinates



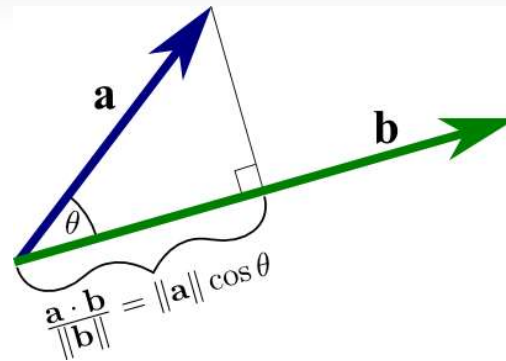
Cross Product



$$\mathbf{c} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$$

$$\mathbf{c} = [(a_2 \times b_3) - (a_3 \times b_2)] \mathbf{i} + [(a_3 \times b_1) - (a_1 \times b_3)] \mathbf{j} + [(a_1 \times b_2) - (a_2 \times b_1)] \mathbf{k}$$

Dot Product

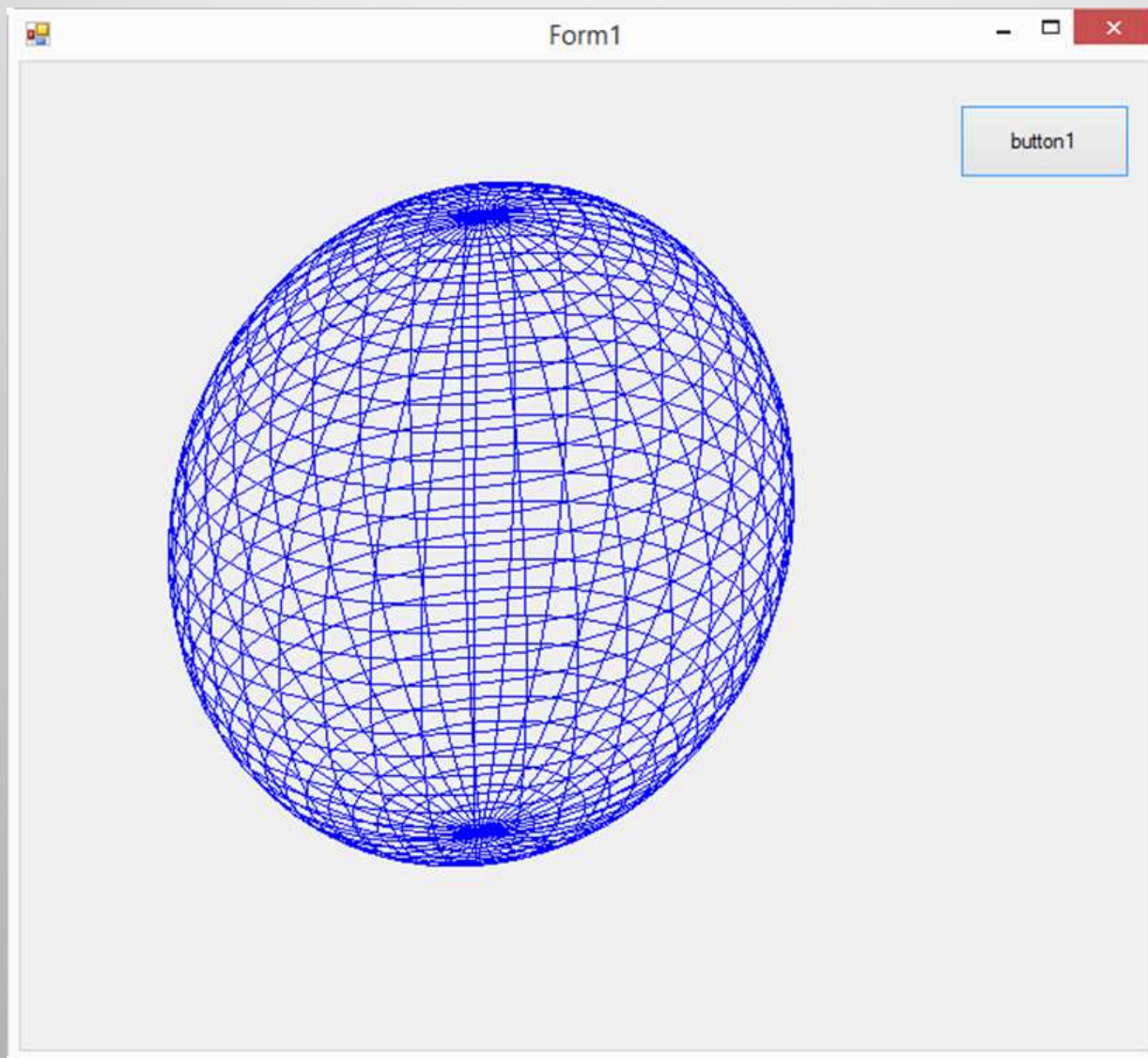


$$\vec{A} \cdot \vec{B} = A_x B_x + A_y B_y + A_z B_z \quad \text{where}$$

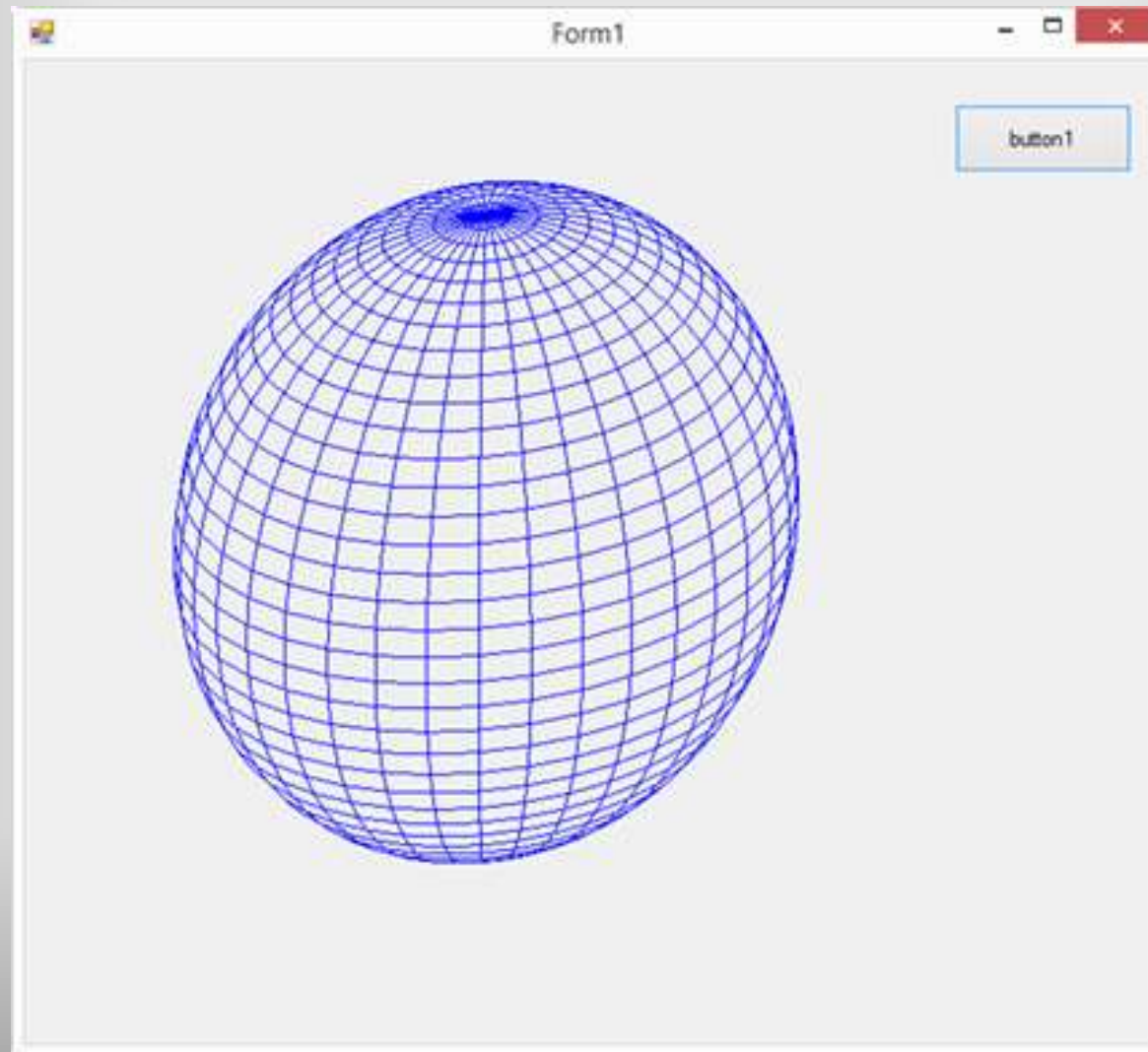
Applications

$$\vec{A} = A_x \vec{i} + A_y \vec{j} + A_z \vec{k}$$
$$\vec{B} = B_x \vec{i} + B_y \vec{j} + B_z \vec{k}$$

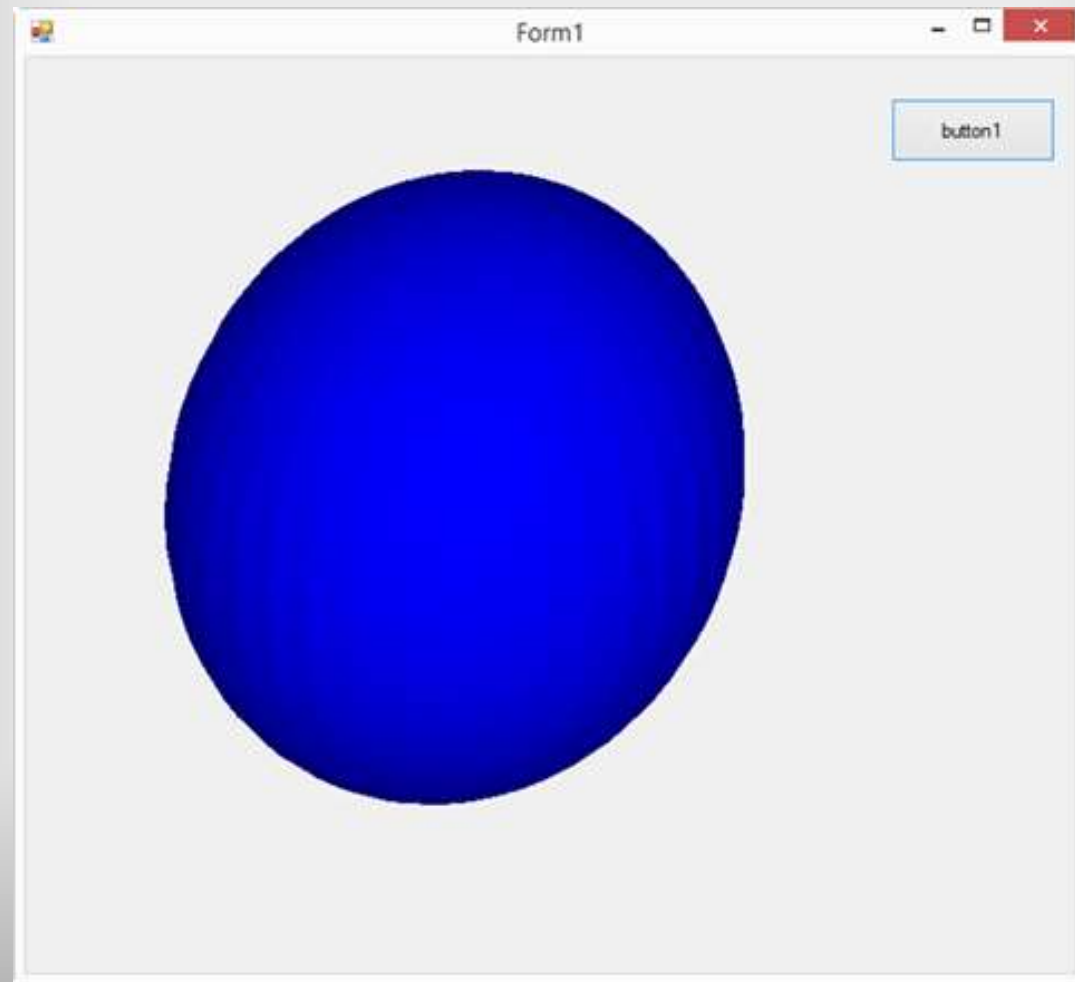
No Culling or Shading



Backface Culling applied

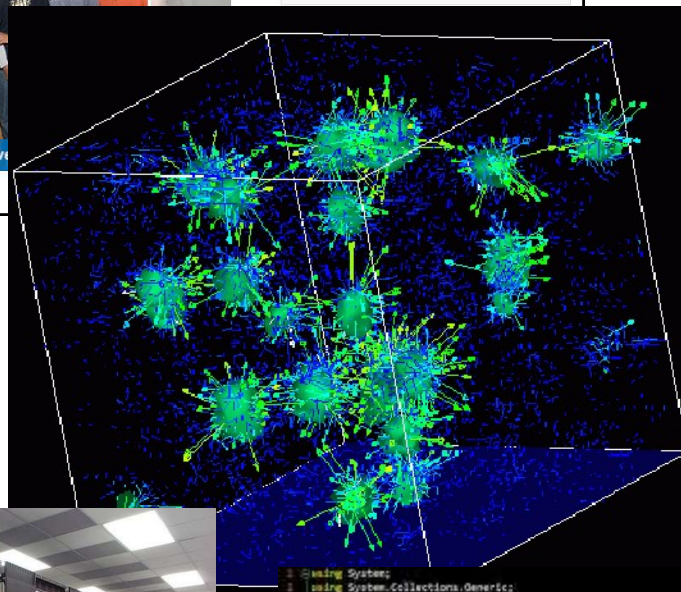
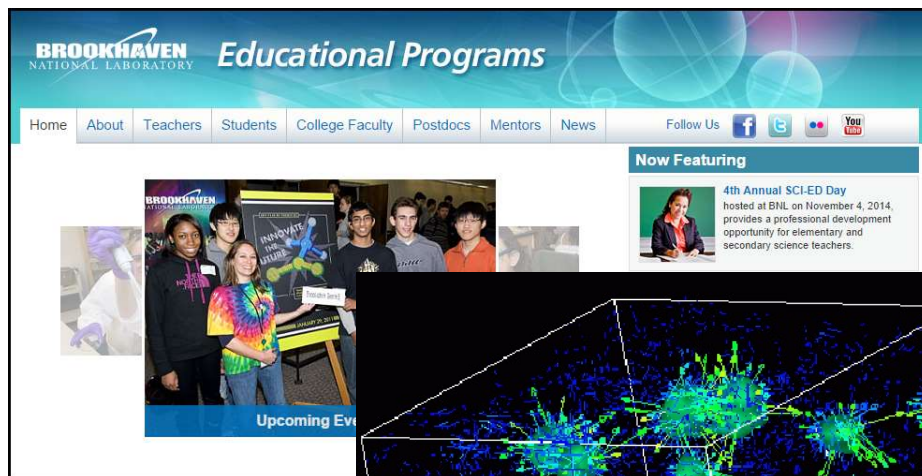


Backface Culling and Facet Shading



What we learned

- We discovered how to represent a 3D image (like a sphere) on a 2D plane (the computer screen)
- How to apply mathematical tools; in this case vector algebra; in a computer program to solve problems
- How to use a collection of rectangles to simulate an object with smooth curves



```

1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Windows.Forms;
9
10 namespace SimpleEvents
11 {
12     public partial class Form1 : Form
13     {
14         Person person = new Person();
15
16         public Form1()
17         {
18             InitializeComponent();
19             person.FirstName = "Christian";
20             person.LastName = "Foss";
21         }
22
23         private void button1_Click(object sender, EventArgs e)
24         {
25             person.HairColor = textBox1.Text;
26         }
27     }
28 }

```

Introducing Scientific Computing

Part 7 The Longest Gene

What is the Longest Repeated Substring?

input string

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| a | a | c | a | a | g | t | t | t | a | c | a | a | g | c |

Step 1 - Form the Suffixes array

suffixes

| | | | | | | | | | | | | | | | |
|----|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | a | a | c | a | a | g | t | t | t | a | c | a | a | g | c |
| 1 | a | c | a | a | g | t | t | t | a | c | a | a | g | c | |
| 2 | c | a | a | g | t | t | t | a | c | a | a | g | c | | |
| 3 | a | a | g | t | t | t | a | c | a | a | g | c | | | |
| 4 | a | g | t | t | t | a | c | a | a | g | c | | | | |
| 5 | g | t | t | t | a | c | a | a | g | c | | | | | |
| 6 | t | t | t | a | c | a | a | g | c | | | | | | |
| 7 | t | t | a | c | a | a | g | c | | | | | | | |
| 8 | t | a | c | a | a | g | c | | | | | | | | |
| 9 | a | c | a | a | g | c | | | | | | | | | |
| 10 | c | a | a | g | c | | | | | | | | | | |
| 11 | a | a | g | c | | | | | | | | | | | |
| 12 | a | g | c | | | | | | | | | | | | |
| 13 | g | c | | | | | | | | | | | | | |
| 14 | c | | | | | | | | | | | | | | |

Step 2 – Sort the Suffixes array

sorted suffixes

| | |
|----|-------------------------------|
| 0 | a a c a a g t t t a c a a g c |
| 11 | a a g c |
| 3 | a a g t t t a c a a g c |
| 9 | a c a a g c |
| 1 | a c a a g t t t a c a a g c |
| 12 | a g c |
| 4 | a g t t t a c a a g c |
| 14 | c |
| 10 | c a a g c |
| 2 | c a a g t t t a c a a g c |
| 13 | g c |
| 5 | g t t t a c a a g c |
| 8 | t a c a a g c |
| 7 | t t a c a a g c |
| 6 | t t t a c a a g c |

Step 3 – Scan the Suffixes array

| sorted suffixes | |
|-----------------|-------------------------------|
| 0 | a a c a a g t t t a c a a g c |
| 11 | a a g c |
| 3 | a a g t t t a c a a g c |
| 9 | a c a a g c |
| 1 | a c a a g t t t a c a a g c |
| 12 | a g c |
| 4 | a g t t t a c a a g c |
| 14 | c |
| 10 | c a a g c |
| 2 | c a a g t t t a c a a g c |
| 13 | g c |
| 5 | g t t t a c a a g c |
| 8 | t a c a a g c |
| 7 | t t a c a a g c |
| 6 | t t t a c a a g c |

Step 3 – Scan the Suffixes array

| sorted suffixes | |
|-----------------|-------------------------------|
| 0 | a a c a a g t t t a c a a g c |
| 11 | a a g c |
| 3 | a a g t t t a c a a g c |
| 9 | a c a a g c |
| 1 | a c a a g t t t a c a a g c |
| 12 | a g c |
| 4 | a g t t t a c a a g c |
| 14 | c |
| 10 | c a a g c |
| 2 | c a a g t t t a c a a g c |
| 13 | g c |
| 5 | g t t t a c a a g c |
| 8 | t a c a a g c |
| 7 | t t a c a a g c |
| 6 | t t t a c a a g c |

Step 3 – Scan the Suffixes array

| sorted suffixes | |
|-----------------|-------------------------------|
| 0 | a a c a a g t t t a c a a g c |
| 11 | a a g c |
| 3 | a a g t t t a c a a g c |
| 9 | a c a a g c |
| 1 | a c a a g t t t a c a a g c |
| 12 | a g c |
| 4 | a g t t t a c a a g c |
| 14 | c |
| 10 | c a a g c |
| 2 | c a a g t t t a c a a g c |
| 13 | g c |
| 5 | g t t t a c a a g c |
| 8 | t a c a a g c |
| 7 | t t a c a a g c |
| 6 | t t t a c a a g c |

Step 3 – Scan the Suffixes array

| sorted suffixes | |
|-----------------|-------------------------------|
| 0 | a a c a a g t t t a c a a g c |
| 11 | a a g c |
| 3 | a a g t t t a c a a g c |
| 9 | a c a a g c |
| 1 | a c a a g t t t a c a a g c |
| 12 | a g c |
| 4 | a g t t t a c a a g c |
| 14 | c |
| 10 | c a a g c |
| 2 | c a a g t t t a c a a g c |
| 13 | g c |
| 5 | g t t t a c a a g c |
| 8 | t a c a a g c |
| 7 | t t a c a a g c |
| 6 | t t t a c a a g c |

Step 3 – Scan the Suffixes array

| sorted suffixes | |
|-----------------|-------------------------------|
| 0 | a a c a a g t t t a c a a g c |
| 11 | a a g c |
| 3 | a a g t t t a c a a g c |
| 9 | a c a a g c |
| 1 | a c a a g t t t a c a a g c |
| 12 | a g c |
| 4 | a g t t t a c a a g c |
| 14 | c |
| 10 | c a a g c |
| 2 | c a a g t t t a c a a g c |
| 13 | g c |
| 5 | g t t t a c a a g c |
| 8 | t a c a a g c |
| 7 | t t a c a a g c |
| 6 | t t t a c a a g c |

Step 3 – Scan the Suffixes array

| sorted suffixes | |
|-----------------|-------------------------------|
| 0 | a a c a a g t t t a c a a g c |
| 11 | a a g c |
| 3 | a a g t t t a c a a g c |
| 9 | a c a a g c |
| 1 | a c a a g t t t a c a a g c |
| 12 | a g c |
| 4 | a g t t t a c a a g c |
| 14 | c |
| 10 | c a a g c |
| 2 | c a a g t t t a c a a g c |
| 13 | g c |
| 5 | g t t t a c a a g c |
| 8 | t a c a a g c |
| 7 | t t a c a a g c |
| 6 | t t t a c a a g c |

Step 3 – Scan the Suffixes array

sorted suffixes

| | |
|----|-------------------------------|
| 0 | a a c a a g t t t a c a a g c |
| 11 | a a g c |
| 3 | a a g t t t a c a a g c |
| 9 | a c a a g c |
| 1 | a c a a g t t t a c a a g c |
| 12 | a g c |
| 4 | a g t t t a c a |
| 14 | c |
| 10 | c a a g c |
| 2 | c a a g t t t a c a a g c |
| 13 | g c |
| 5 | g t t t a c a a g c |
| 8 | t a c a a g c |
| 7 | t t a c a a g c |
| 6 | t t t a c a a g c |

longest repeated substring

1 9

a a c a a g t t t a c a a g c

What is the Longest Repeated Substring?

input string

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| a | a | c | a | a | g | t | t | t | a | c | a | a | g | c |

file:///D:/DaveB/SciComp 101 - Instructor/Class 29 - Longest Repeated Substring/Lon

The longest recurring substring is: acaag

```

string dna = String.Empty;
dna += "atgcgcaactcaaccccttactactcccgccagatatgaatccctccgcttttaggaataac";
dna += "ggacggcatggactttcatcgtctccttgggttttacctgatggacagctctactcggtc";
dna += "cggctactcagaacaagtgcttaccaaagaatacatgaatgcccaagtaacctggggttca";
dna += "attcacgtagccaacgacgcatgctggttcaggatcgcgagatataagcagcagcgaacg";
dna += "gttaccgacatccatacatcgggtctcttgctatacatatgagtctcgagcatcgtagag";
dna += "acttgggagccgtctcaagcacagttcattaggagtcttgaatgggattgtttgttccc";
dna += "tcaaaaggagtctgcctattgtgcaatgacctgtcgagatccgacagaacgcagataagg";
dna += "gtgtcaagtggtcgagaggtgcgtataaaatcatttcaatatcgcgaggccctgaccat";
dna += "aaattctcgtacaatcaccatagatcaaatatggggtatcccttgagcgaggggggtg";
dna += "gggccttttaagtctactctttaactacaacagctcctacgtagcaaggcttagccgtcac";
dna += "atctattatagaccgtggttccatgacactaggagtttctagggtgagcagtcagcagc";
dna += "tggcgtcgttcacttaatagacagtttcgagtcaacacgtcgcgcatctcatatatgta";
dna += "aaggaaagcaatcaccataaggcccaacgtacgtcttacctaagagacacttaagtttt";
dna += "agactgcccgtagtagcgaacggccttagtgacgatacgaattcaatcgaatgctg";
dna += "agcgcaaaagtacgaaggttgatcgccacgcaagtaacaatacttttatgatctgca";
dna += "gtatacacactcctcgaccgggggacacgacactcgtagcttctcctgaccaattgtct";
dna += "gtggagagtcagacgtggtttataacgcaggtcttccattcccacagtcgagctggcc";
dna += "tgtacttgggaaggtctccgagagctctttgtgaggttgagtagatgtccaagctcacgc";
dna += "atccgcatactttgttatggtctgacggcccgaggatcggtctctacggattatccaa";
dna += "tacgccccctaggaactataatgaacgtagtctacgacgcccaggggtgtagacggatc";
dna += "ctcgtaaaagactggtgaaaaggacttatgtagcaacgggtgttcacacgggtgcagaggtt";
dna += "gaggacgggaacataatttattttaaccagcaagtcaccttctttgtggagttttggtac";
dna += "caatcaaatggctttgcatgccgaatcgcccttttcgacccgaggacgcggcgggcgta";
dna += "cgatccgtgtcaaattttcgggaaacccgattcactcaagaactacatgatataaaggcg";
dna += "tgtacggtagctaactcccaagtcattaccttgggtgtccagcgtactctcagtagacca";
dna += "tacaaactcgttactcagcggaggtggtgcggacgctcctacgggaataccttgcaaaagc";
dna += "gcacggcctctccactcctcacgtttcagccagacactaaaaataccggttcgctacagc";
dna += "tcacatcgttactcttatactcttattaaaccagcacatcccttcggtctgacgcagccg";
dna += "ccagtgtctcgcgcaattacctcacgggtgaaaagcacagagcgccggcgactgcttate";
dna += "ctacgagagaacaaggtttatttctacctaaccgcctggagccacatggggcggtcgc";
dna += "ctcgcgactccgattcgtacggcggaactgcacccatcatcgtaggaccagaacctataat";
dna += "gttccctcgtctgaaagccggccaagctcgaggggtgagcacaacggttacagacggcac";
dna += "gaattattaggccagcgtgttttaaaatttgcaaaacaggtggctcttgacatacgaat";
dna += "gtgcatacacagggggtgcaggggcttgacatggagacagaatggggacgcggtctaaa";
dna += "gaccatgcctgttctcaacggcaccgcataaaagctgcctcaagcttatcaaaactttat";
dna += "gaatcctttcgcccacgccaacaacaggttttcaggtagcatatcctctaaatctggc";
dna += "ttctacactcgctttaaaacctgtcgaactcagttctggagtctcaagcaacgtaactca";
dna += "tcacgcacagcttaccagtgatcgtaattcgcgctgctagactcacgagcacaactagt";
dna += "gtcgaggatgtgtccgcaatgccgctccaggatgagggctgtctcgatgccaaggcc";
dna += "cggggaatttctggagctaacagaataaaggtaaaaaacgagtggtcctaccttcaaatc";
dna += "tctccatacgccctccactcgtcggattgagaagacgtcgctacctctcgttgg";
dna += "ctgtcttatgtccgggtataggccaaggaccgttaagacagacgacccgtctacggac";
dna += "ttaagtcgaccgtacactagagtgaacgatagcgcgctggaggcgtcgatctttttcg";
dna += "caaccaataacgttttccaaatcaggccactactgaggatgagcagggcgtgggtgt";
dna += "cggacaggaggctcggtcacctggctccaggaaaggagcgtcattgcaccagactaatc";
dna += "caggaacagcgcgctcctgaaagcgcaaggttgacagattgtctggcggtgcagaa";
dna += "tcacgggtgtctcgtattacttgggcttcagtaccatttgtagctgctgtaaacctcca";
dna += "ggaaagcggggagccagggcctgaagcaagcagaccgagcagaatacaaaaaatggcct";
dna += "gtcgggtgcatcaaatgatatcccagcggtttacgctgctgcatagcatgacgcgtcgg";
dna += "ttacaatcaactctatatcaacagatcgtagattgaaggcaatcctgtatagcctag";

```

What is the Longest Repeated Substring?

Lab

- Implement the code to compare to suffix strings a and b
- As long as the letters in each string keep matching, we can add them to the longestMatch string
- As soon as they are no longer equal, we bail out and return whatever we've built up so far

```
static string GetLongestMatch(string s1, string s2)
{
    string a = s1;
    string b = s2;

    // Ensure string a is shorter than b
    if (s1.Length > s2.Length)
    {
        a = s2;
        b = s1;
    }

    string longestMatch = String.Empty;

    // Implement the code to compare the characters between string a and b
    // building the longestMatch as long as the characters are the same

    return longestMatch;
}
```



```

string dna = String.Empty;
dna += "atgcgcaactcaaccccttactactcccgccagatatgaatccctccgcttaggaataac";
dna += "ggacggcattggactttcatcgtctccttgggttttacctgatggacagctctactcggtc";
dna += "cggctactcagaacaagtgcttaccaaagaatacatgaatgcccaagtaacctggggttca";
dna += "attcacgtagccaacgacgcatgctggttcaggatcgcgagatataagcagcagcgaacg";
dna += "gttaccgacatccatacatcggctcttggctatacatatgagtcctcgagcatcgtagag";
dna += "acttgggagccgtctcaagcacagttcattaggagtcctgaatgggattgttggttcccg";
dna += "tcaaaaggagtcgtcctattgtgcaatgacctgtcgagatccgacagaacgcagataagg";
dna += "gtgtcaagtgtagaggtgctctataaaatcatttcaatatcgaggaggccctgaccat";
dna += "aaattctcgtacaatcaccatagatcaaatatggggtatcccttgagcgaggggggtg";
dna += "gggccttttaagtcaactctttaactacaacagctcctacgtagcaaggcttagccgtcac";
dna += "atctattatagaccgtgggttccatgacactaggagtttctagggtgtagtcagcagc";
dna += "tggcgtcgttcacttaatagacagtttcgagtcacacgctcgcgatctcatatatgta";
dna += "aaggaaagcaatcaccataaggcccaacgtacgtcttacctaagagacacttaagtttt";
dna += "agactgcccgtagtagcgaaacggccttagtgacgatacgaaaattcaatcgaatgctg";
dna += "agcgcaaaagtacgaagggtgatcgccacgcaagtaacaatacttttatgatcttgc";
dna += "gtatacacactcctcgaccggggggacacgacactcgtagcttctcctgaccaattgtct";
dna += "gtggagagtcagacgtggtttataacgcaggtcttccattcccacagtcgagctggcc";
dna += "tgtacttgggaaggtctccgagagctctttgtgaggttgagtagatgtccaagctcacgc";
dna += "atccgcatactttgttatggtctgacgcccaggatcggtctctacgattatccaa";
dna += "tacgccccctaggaactataatgaacgtagtctacgacgcccaggggtgtagacggatc";
dna += "ctcgtaaaagactgggtgaaaaggacttatgtagcaacgggtgtcacacggtgcagacgttc";
dna += "gaggacgggaacataatatttttaaccagcaagtcaccttctttgtggagtttggtag";
dna += "caatcaaatggctttgcatgccgaatcgcccttttcgacccgaggacgcggcgggcgta";
dna += "cgatccgtgtcaaattttcgggaacccgattcactcaagaactacatgatataaaggcg";
dna += "tgtacggtagctaaccccaatcattaccttgggtctcagcctactctcaatagacca";
dna += "tacaactcgtta";
dna += "gcacggcctctc";
dna += "tcacatcgttact";
dna += "ccagtgtctcgc";
dna += "ctacgagagaaca";
dna += "ctcgcgactccga";
dna += "gttccctcgtctg";
dna += "gaattattaggcc";
dna += "gtgcatacacagg";
dna += "gaccatgcctgtt";
dna += "gaatcctttcgcc";
dna += "ttctacactcgct";
dna += "tcacgcacagtct";
dna += "gctgcaggatgttctcgaaatgcgtctcaggatagggctgtcttcgatgccaggcc";
dna += "cggggaatttctggagctaacagaataaagggtacaaaaacgagtgccctaccttcaaatc";
dna += "tctcccatagccctccatccactgctcggattgagaagacgtcgtacctctccgttgg";
dna += "ctgtcttatgtccgggtataggccaaggaccgttaagacagacgacccgtctacggac";
dna += "ttaagtcgaccgtacactagagtgaacgatagcgcgctggaggcgtcgatcttttttcg";
dna += "caaccaataacgttttccaaatcaggccactactgcaggatagcgaggcgtgggtgt";
dna += "cggacaggaggctcggtcacctggctccaggaaaggagcgtcattgcaccagactaatc";
dna += "caggaacagcgcgctcctgaaagcgcaaggttgacagattgtctggcggtgcagaa";
dna += "tcacgggtgtctcgattacttgggttcagttaccattgtgacgtgctgaaacctcca";
dna += "ggaaagcggggagccagggtcgaagcaagcagaccgagcagaatacaaaaaatggcct";
dna += "gtcgggtgatcaaatgatatcccgagcggttacgctgctgcatagcatgacgcgtcgg";
dna += "ttacaatcaactctatatcaacagtatcgtagattgaaggcaatctcgttatagcctag";

```

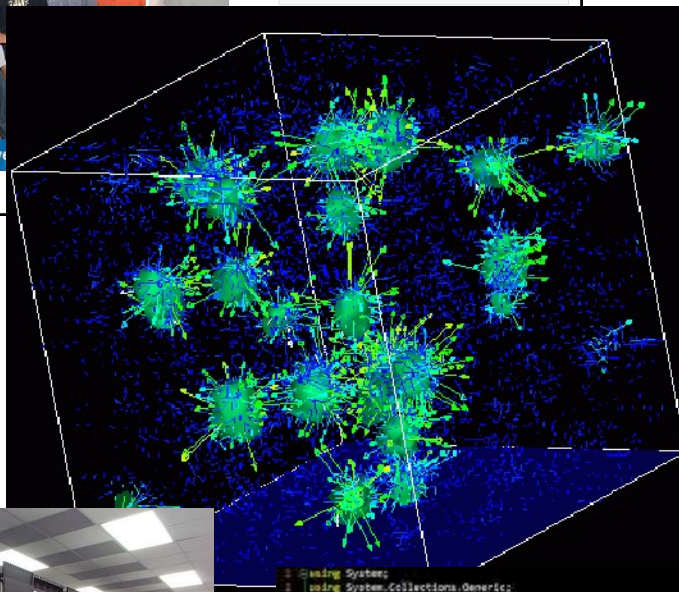
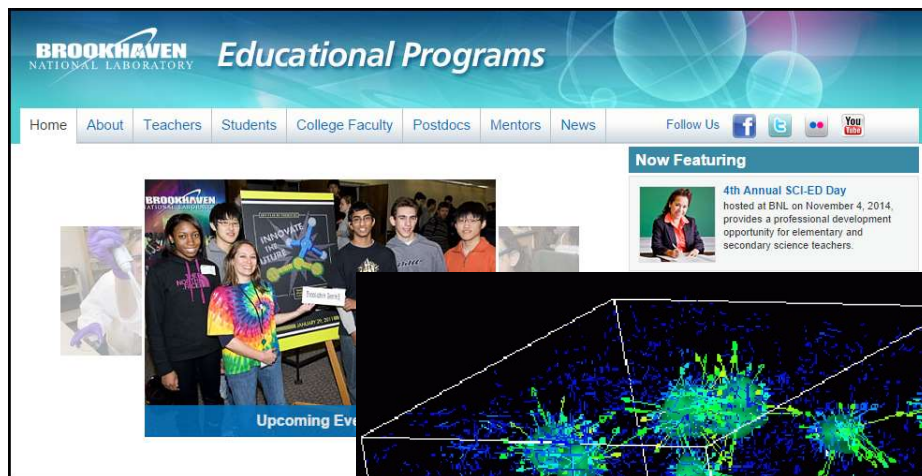
What is the Longest Repeated Substring?

file:///C:/Users/dbiersach/Desktop/SciEd Day/Part 08 - The Longest Gene/LongestRepeatedSubs

The longest recurring substring is: **caatcaccata**

Research Questions

- How could we update the code to match against single (or multiple) “**wildcard**” (unknown/place holder) sequences?
- How could we update the code to produce a **frequency table** to show the *number of times* successively longer substrings are located?
- What might be the biological significance of a **very long substring** that appears only a *few* times within the entire sequence?



```

1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Windows.Forms;
9
10 namespace SimpleEvents
11 {
12     public partial class Form1 : Form
13     {
14         Person person = new Person();
15
16         public Form1()
17         {
18             InitializeComponent();
19             person.FirstName = "Christian";
20             person.LastName = "Foss";
21         }
22
23         private void button1_Click(object sender, EventArgs e)
24         {
25             person.HairColor = textBox1.Text;
26         }
27     }
28 }

```

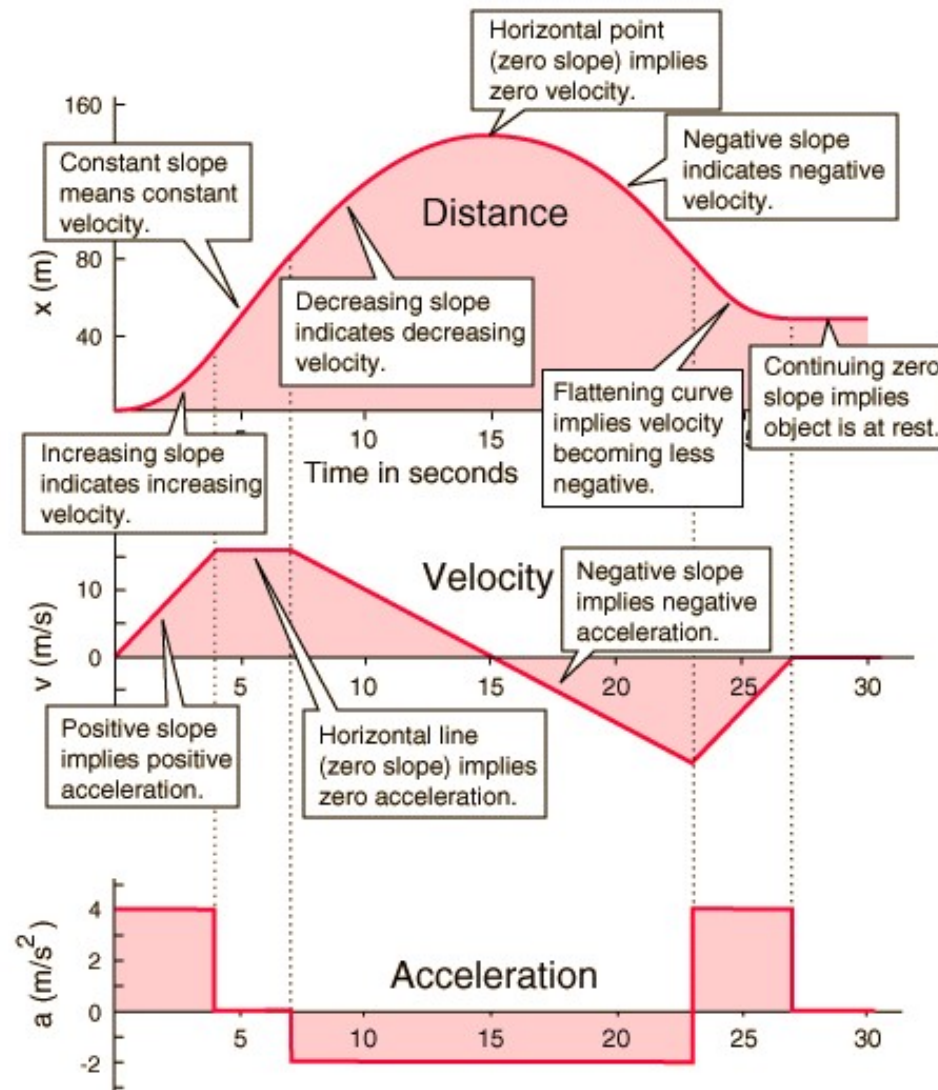
Introducing Scientific Computing

Part 8 Monte Carlo Integration

Why do we need integrals?

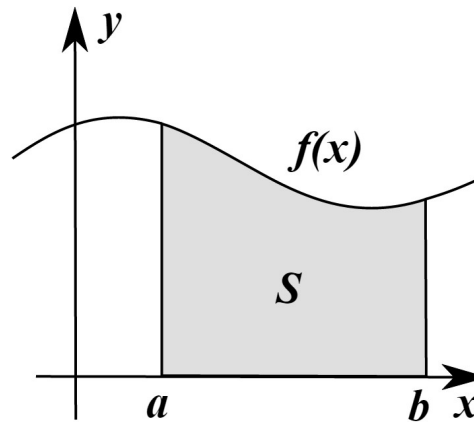
- How to calculate the total change in a variable X
 - When variable X *depends* on the changes in variable Y...
 - ... and variable Y *depends* on the changes in variable Z...
 - ... and variable Z is constantly changing...
- Think about an accelerating car and the total distance it will travel in a given number of seconds
 - The total distance *depends* on the velocity of the car...
 - ... the velocity of the car *depends* on the acceleration
 - ... and the acceleration is constantly changing

Why do we need integrals?



Why do we need integrals?

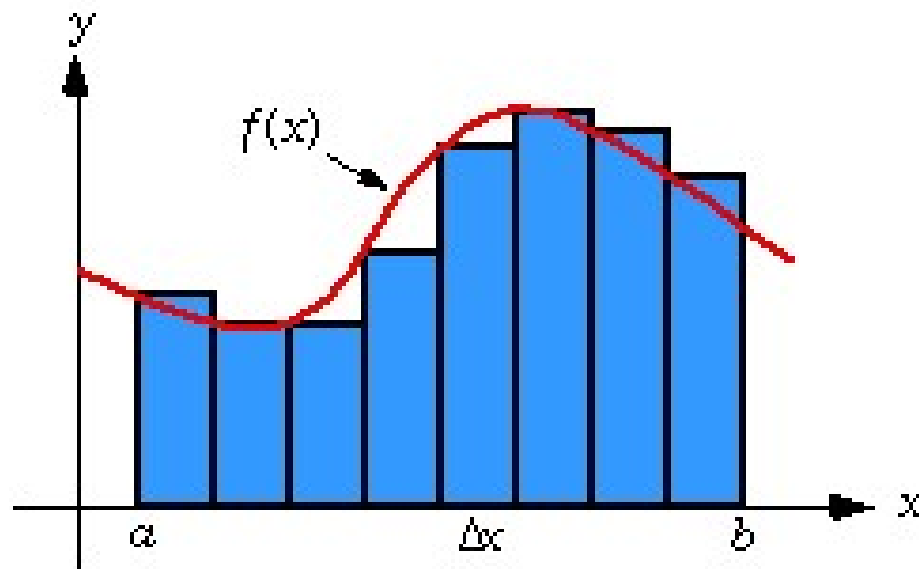
- The **integral** of a function can be defined as the **area under a curve** $f(x)$ within the region $[a,b]$



- There are ways to often determine exactly the value of the integral of $f(x)$ – which we would write $F(x)$
- However, sometimes it is not possible to find an analytic expression for $F(x)$ – so we use **numerical** integration

Riemann Sums

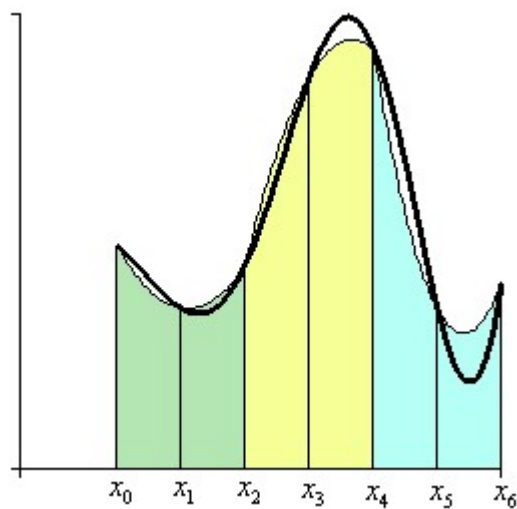
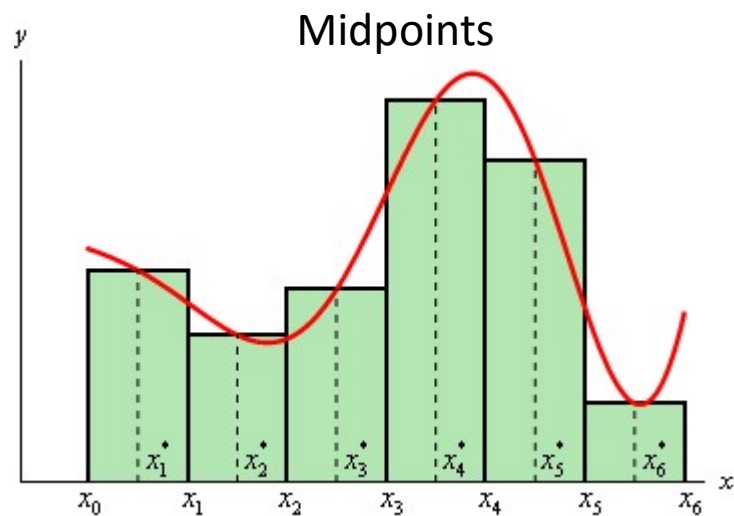
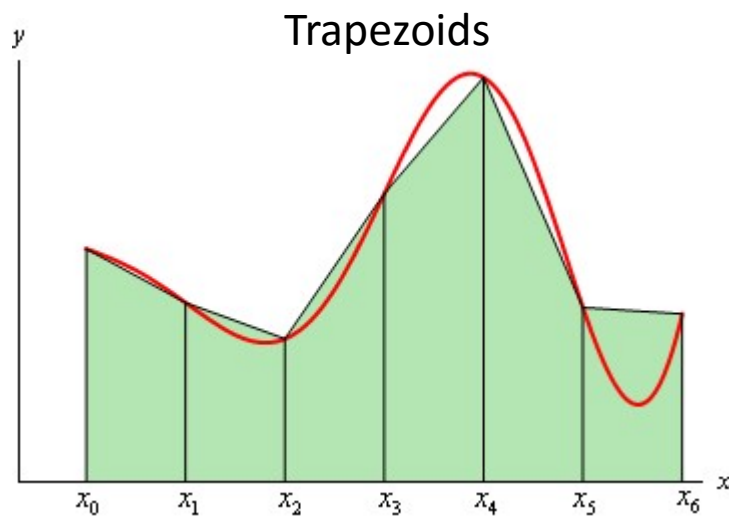
- One way we can integrate $f(x)$ is to divide the area under the curve into strips (**intervals**) and sum the area of each strip
- This estimate may not be totally accurate because we might have **gaps** between the true value of $f(x)$ and the top of a strip



Riemann Sums

- The width of each strip is $\Delta x = \frac{(b-a)}{\# \text{ of intervals}}$
- We can minimize the gaps by increasing the number of intervals, which makes Δx get smaller
- There are different strategies for determine the shape and height of each strip
 - Trapezoids
 - Midpoints
 - Parabolas (Simpson's Rule)
- Depending upon the particular shape of $f(x)$, one method might be more accurate than the other two

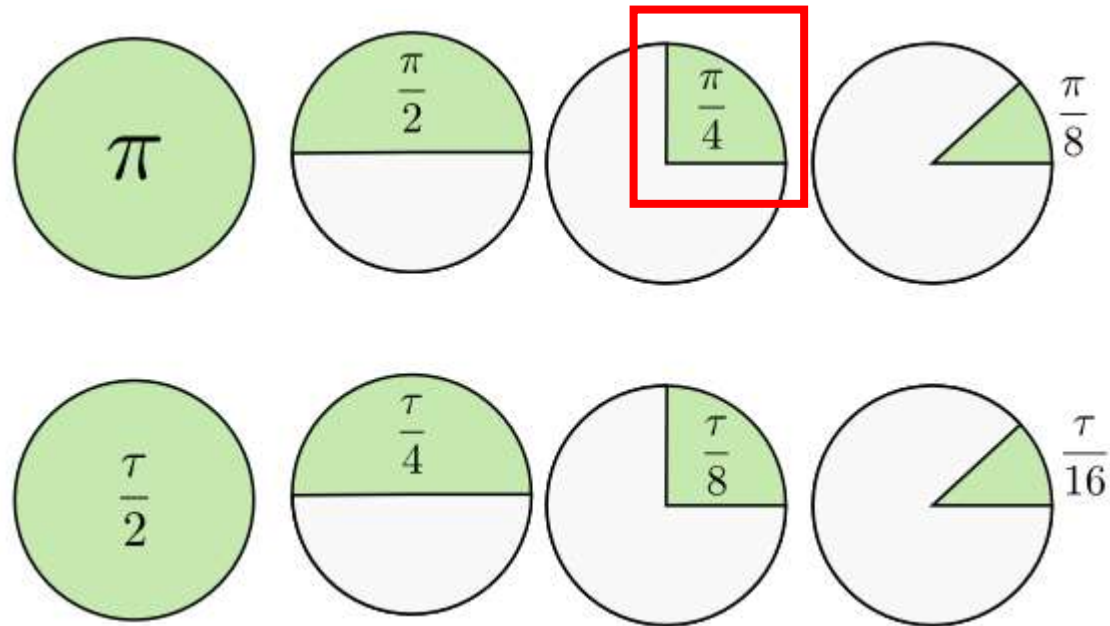
Riemann Sums



Parabolas
(Simpson's Rule)

Let's experiment with Numerical Integration

- We want to calculate the area under the first quadrant of the unit circle (≈ 0.785398163)

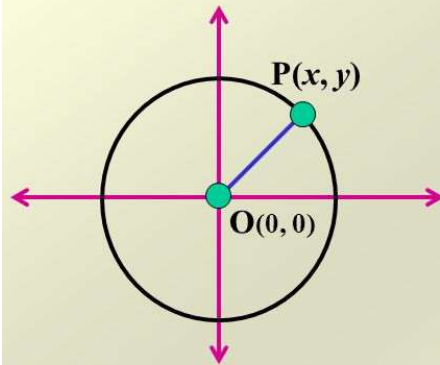


Let's experiment with Numerical Integration

- For a unit circle at the origin: $f(x) = \sqrt{1 - x^2}$

Developing the Standard Forms of the Equation of a Circle

Note: OP is the radius of the circle.



$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} = d_{OP}$$

$$\sqrt{(x - 0)^2 + (y - 0)^2} = r$$

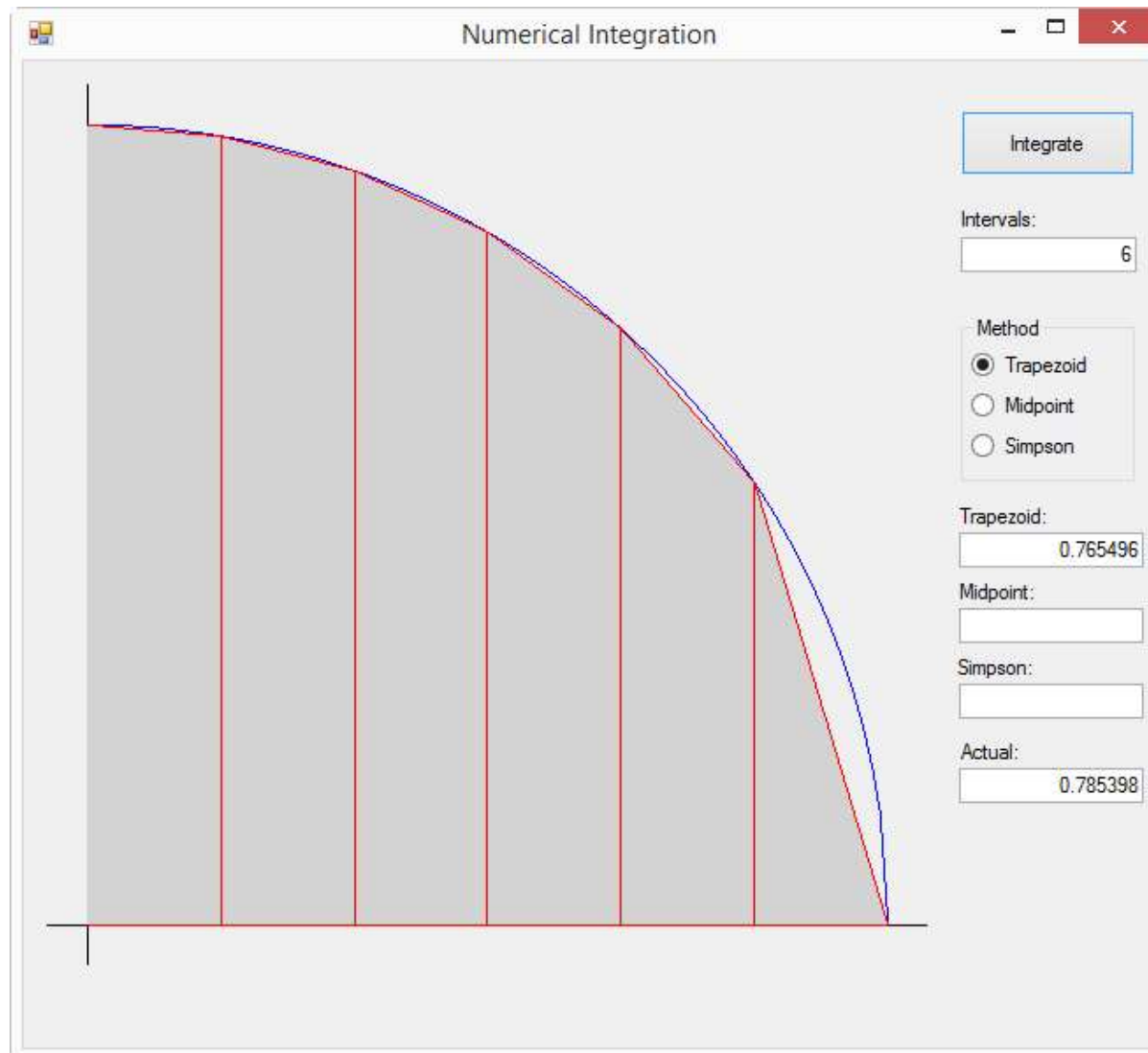
$$\sqrt{x^2 + y^2} = r$$

$$x^2 + y^2 = r^2$$

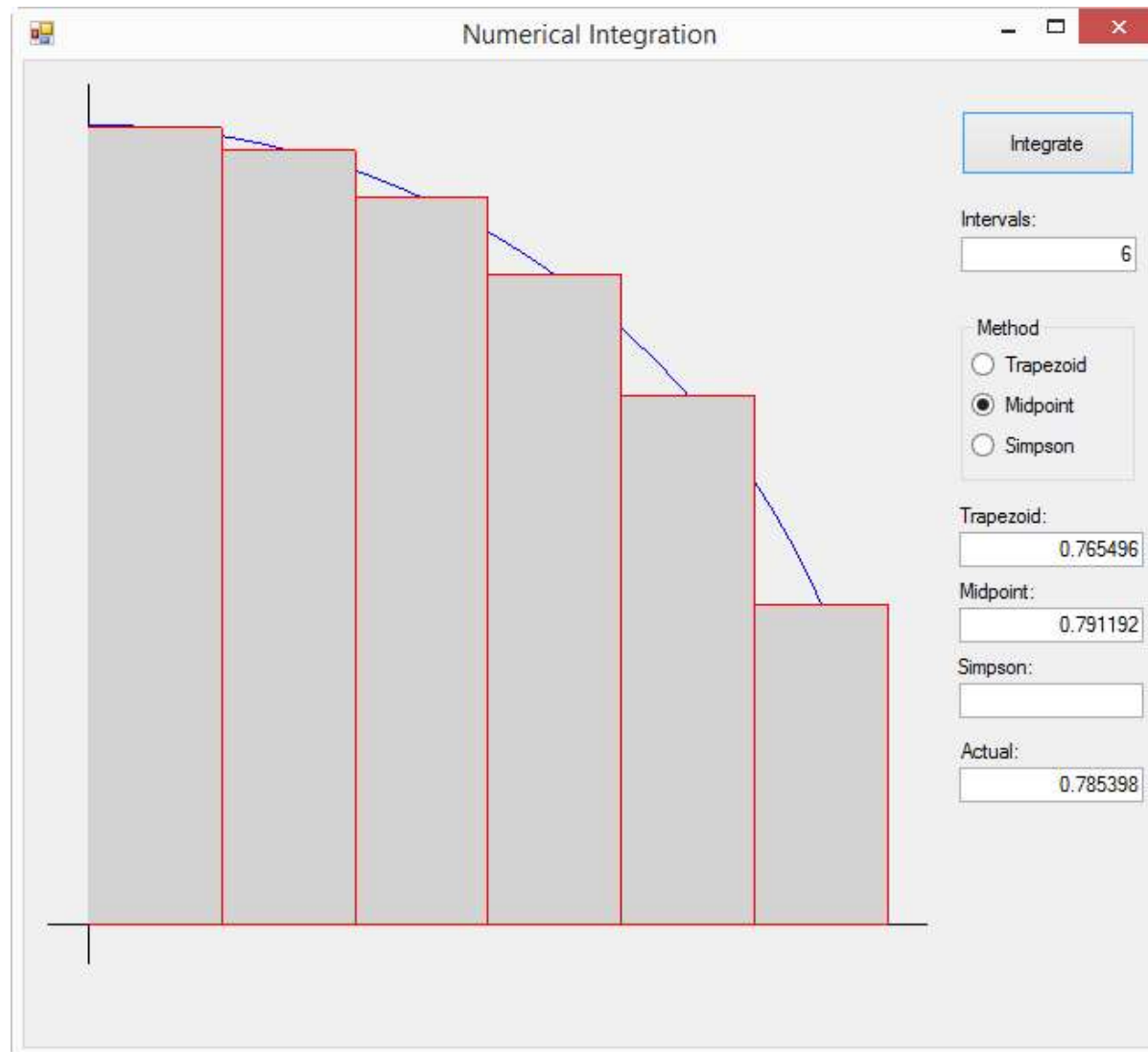
The standard form of the equation of a circle with its centre at the origin (0, 0) is $x^2 + y^2 = r^2$.

3.3.2

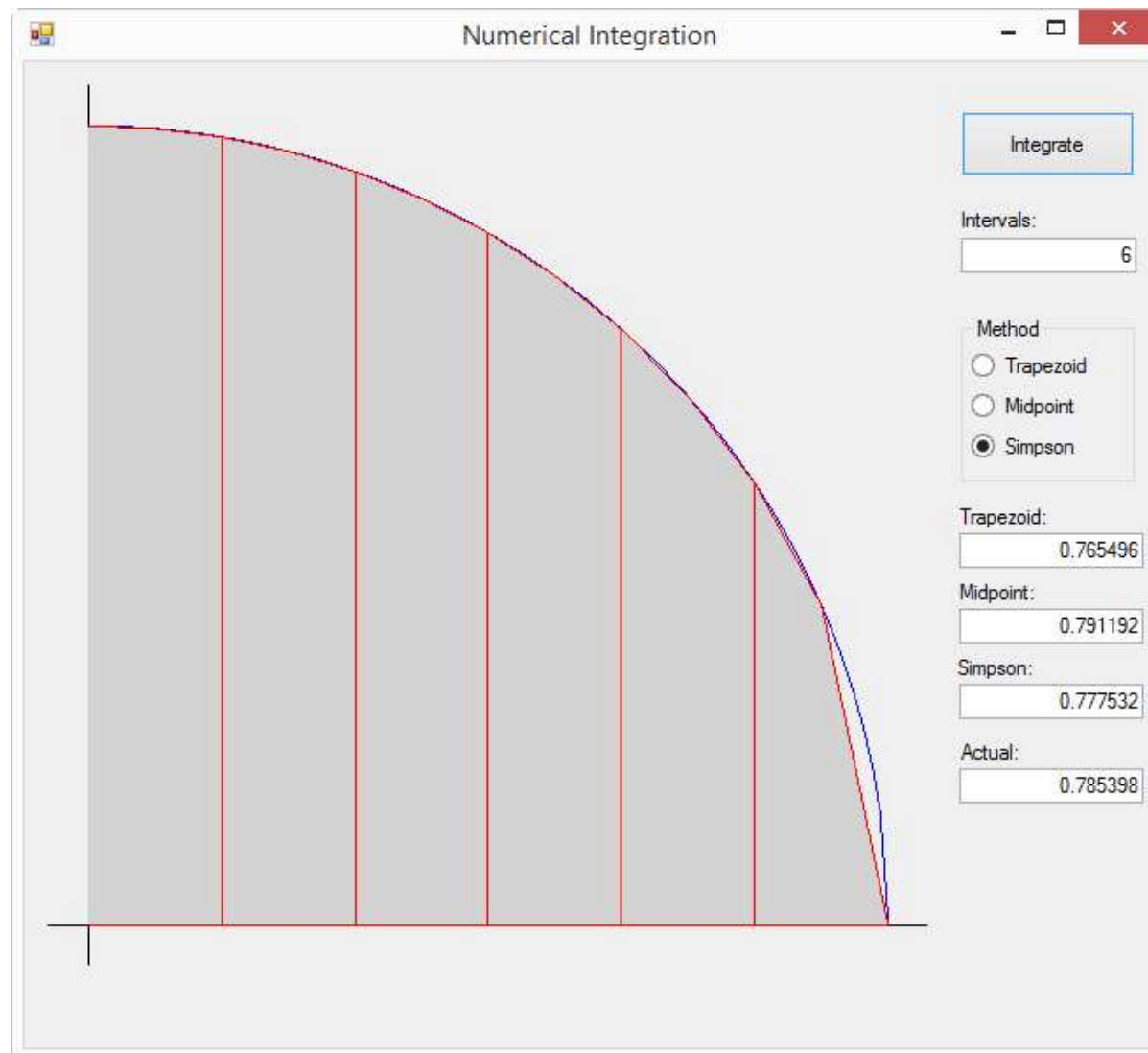
Riemann Integration - Trapezoid



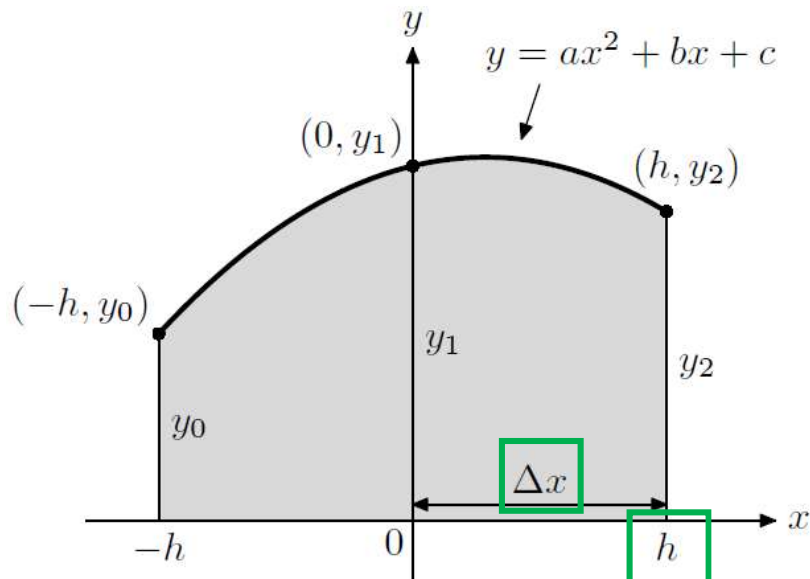
Riemann Integration - Midpoint



Riemann Integration – Simpson



Simpson's Rule is more accurate!



$$y_0 = ah^2 - bh + c$$

$$y_1 = c$$

$$y_2 = ah^2 + bh + c$$

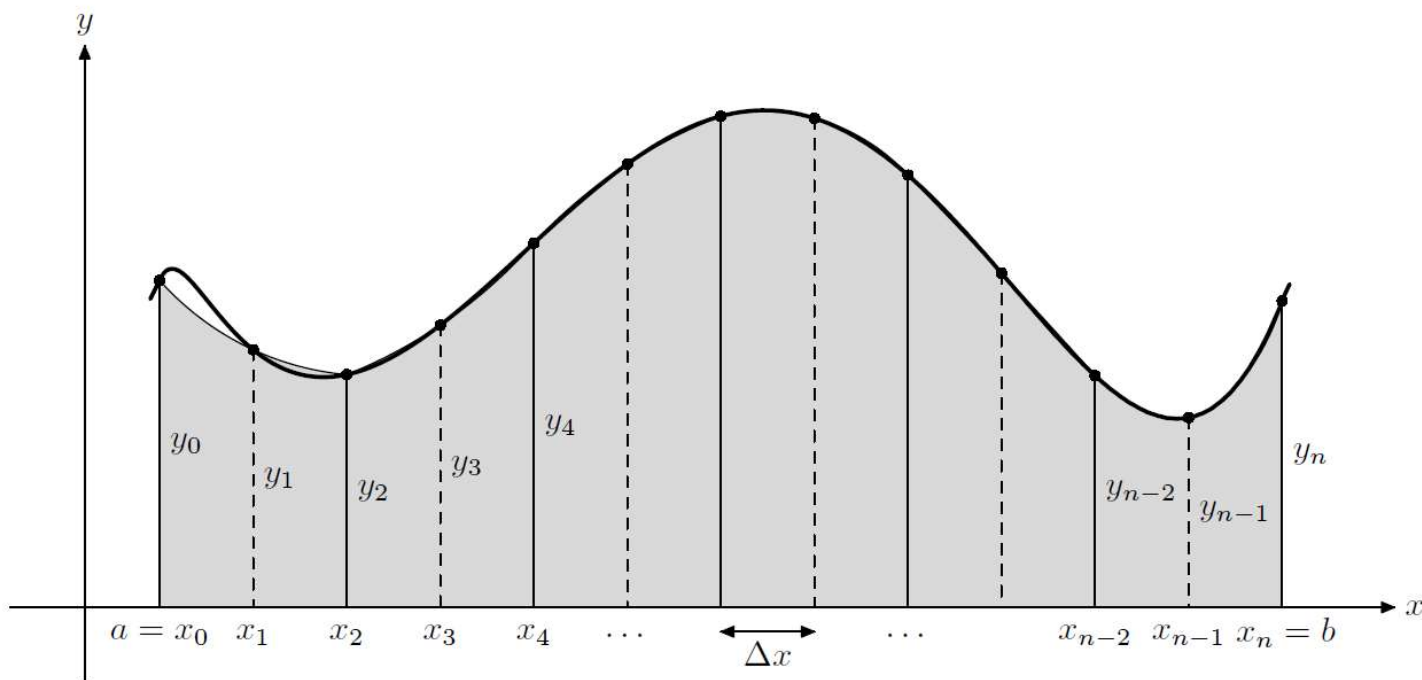
$$y_0 + 4y_1 + y_2 = (ah^2 - bh + c) + 4c + (ah^2 + bh + c) = 2ah^2 + 6c$$

$$\begin{aligned} A &= \int_{-h}^h (ax^2 + bx + c) dx \\ &= \left(\frac{ax^3}{3} + \frac{bx^2}{2} + cx \right) \Big|_{-h}^h \\ &= \frac{2ah^3}{3} + 2ch \\ &= \frac{h}{3} (2ah^2 + 6c) \end{aligned}$$

$$A = \frac{h}{3} (y_0 + 4y_1 + y_2) = \frac{\Delta x}{3} (y_0 + 4y_1 + y_2)$$

Simpson's Rule is more accurate!

$$y_0 = f(x_0), \quad y_1 = f(x_1), \quad y_2 = f(x_2), \quad \dots, \quad y_n = f(x_n).$$



$$\int_a^b f(x) dx \approx \frac{\Delta x}{3} (y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + \dots + 4y_{n-1} + y_n)$$

Simpson's Rule is more accurate!

$$\int_a^b f(x) dx \approx \frac{\Delta x}{3} (y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + \cdots + 4y_{n-1} + y_n)$$

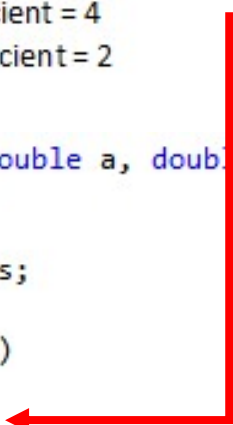
| Point | y0 | y1 | y2 | y3 | y4 | y5 | y6 |
|-------|----|----|----|----|----|----|----|
| Coeff | 1 | 4 | 2 | 4 | 2 | 4 | 1 |

The first and last point have coefficient = 1

Every point with an odd index has coefficient = 4

Every point with an even index has coefficient = 2

```
private double IntegrateUsingSimpson(double a, double b, int intervals)
{
    double sum = f(a) + f(b);
    double deltaX = (b - a) / intervals;
    a += deltaX;
    for (int i = 1; i < intervals; i++)
    {
        int coeff = 2 * (i % 2 + 1);
        sum += coeff * f(a);
        a += deltaX;
    }
    double integral = (deltaX / 3) * sum;
    return integral;
}
```



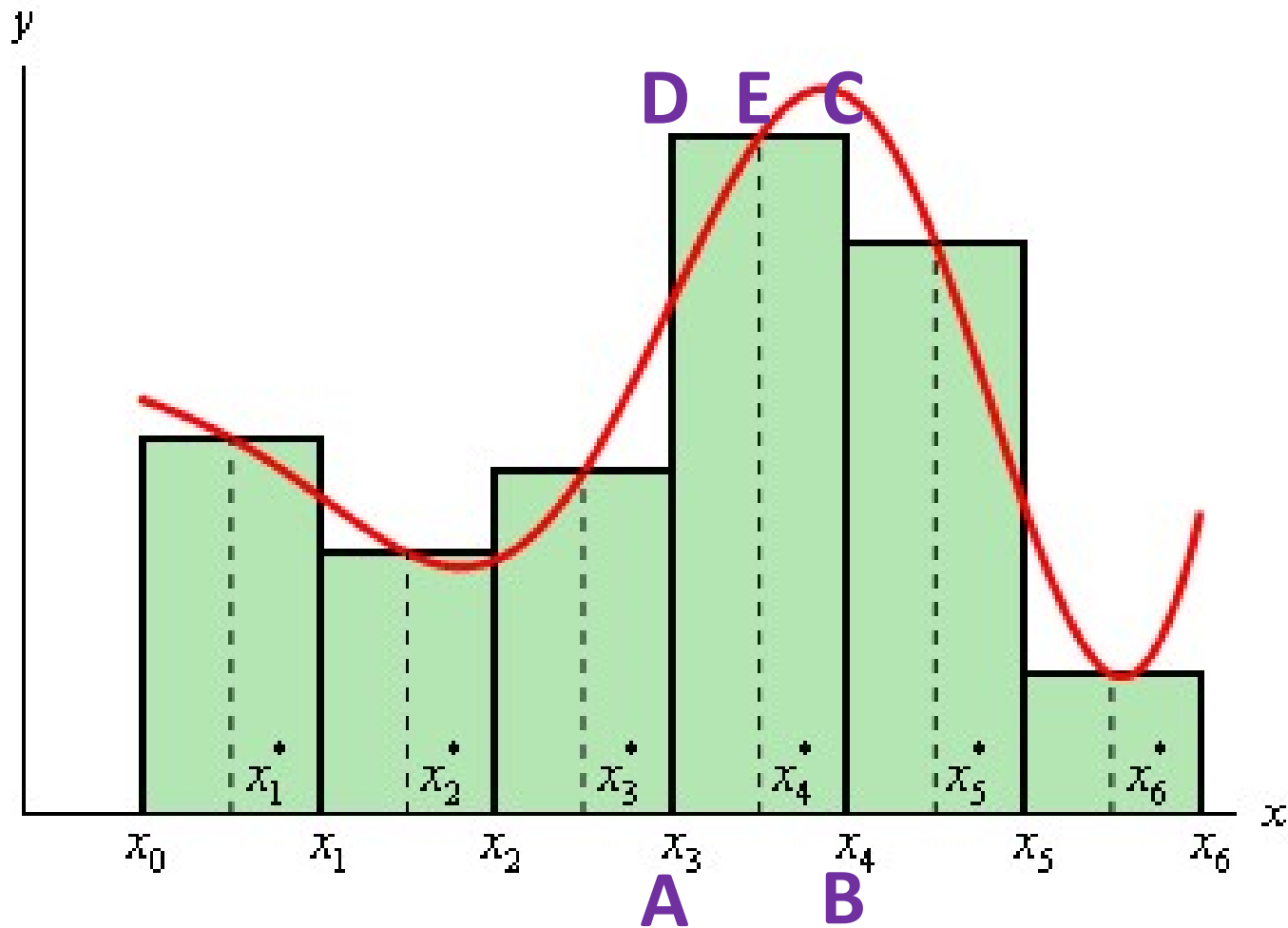
Let's experiment with Numerical Integration

- What happens to the accuracy as you increase the number intervals?
- Is there a law of diminishing returns as you continue to increase the number of intervals?
- You can change the expression for $f(x)$ to investigate other curves

```
private double f(double x)
{
    return Math.Sqrt(1 - x * x);
}
```

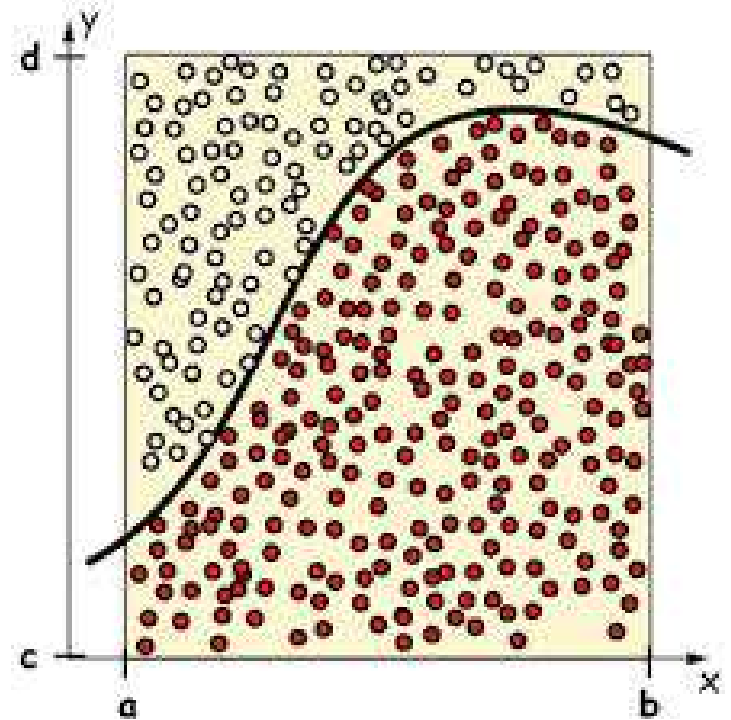
```
// Define WORLD boundary values
double worldXmin = -.05;
double worldYmin = -.05;
double worldXmax = 1.05;
double worldYmax = 1.05;
```


Code points on each strip



The Monte Carlo Method

- When we need to take integrals in higher dimensions, Riemann sums may get unwieldy
- With Monte Carlo, we randomly sample points across the entire space and count how many are below the curve
- The ratio between points below the curve to total points is an estimate of the integral
- Monte Carlo is a non-deterministic approach, versus Trapezoidal, Simpson's etc.



The Monte Carlo Method

Monte Carlo approximation



Ulam

1940's
→ Los
Alamos

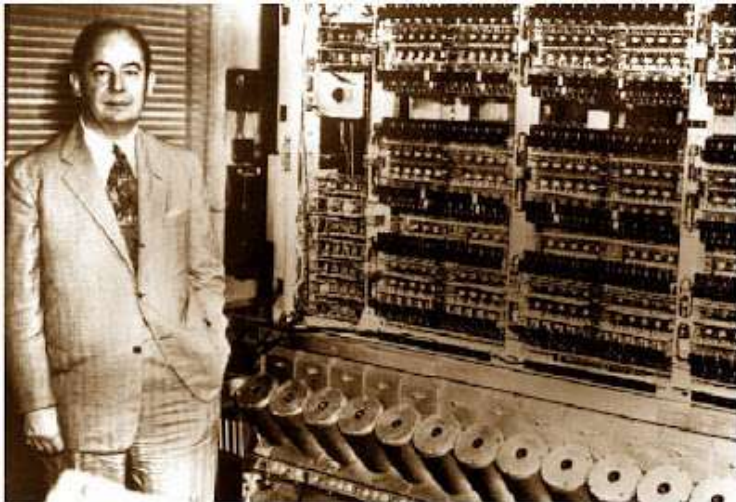


Fermi



Von Neumann

1930's

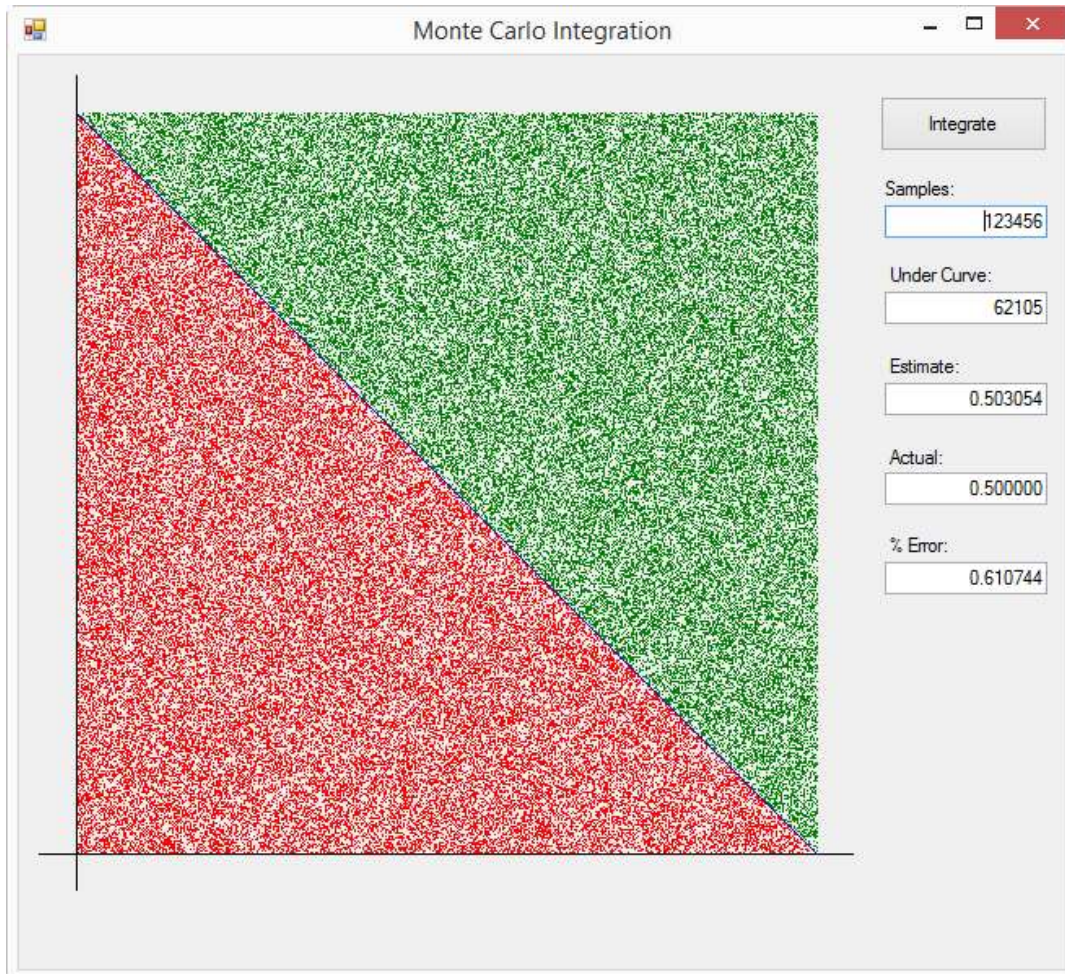


Johnny von Neumann [1903-1957] alongside the Maniac computer at the Institute for Advanced Studies, Princeton.



Monaco

Monte Carlo Integration



$$f(x) = 1 - x$$

Area for a triangle

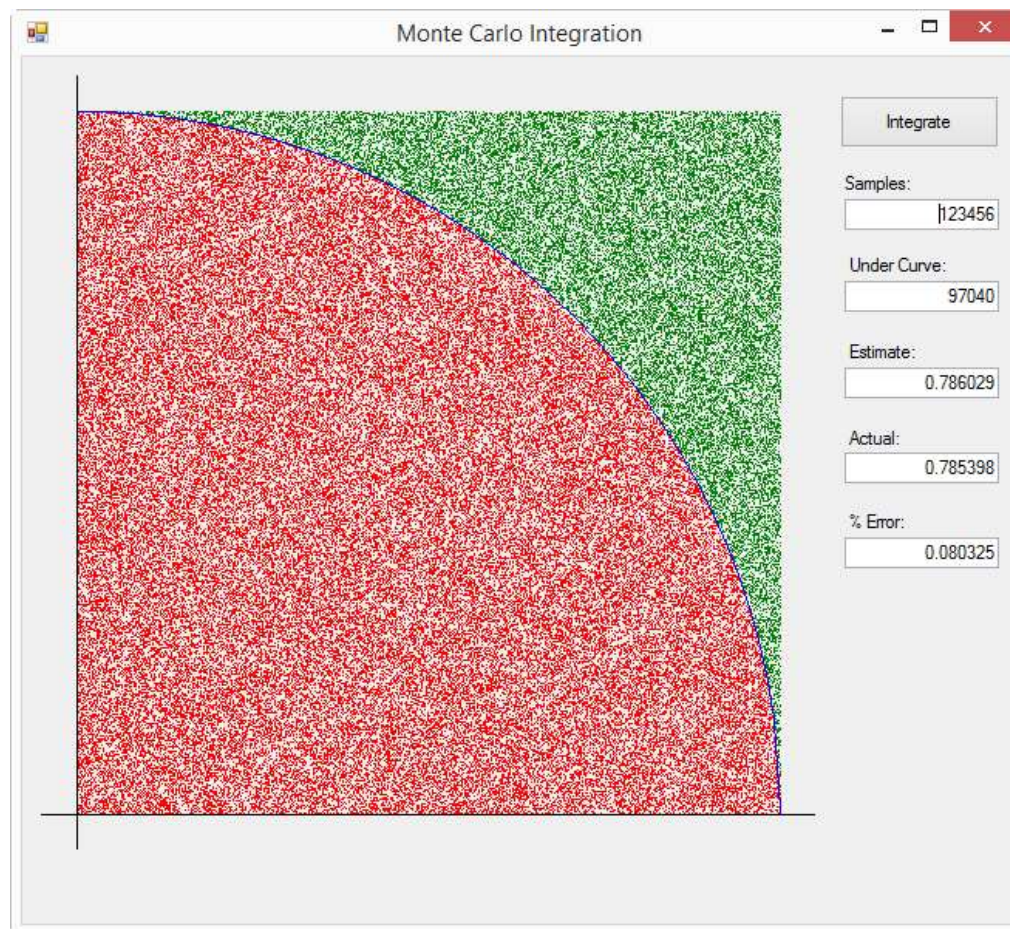
$$b = 1 \text{ and } h = 1$$

$$A = \frac{b \cdot h}{2}$$

$$A = \frac{1}{2}$$

Monte Carlo Integration

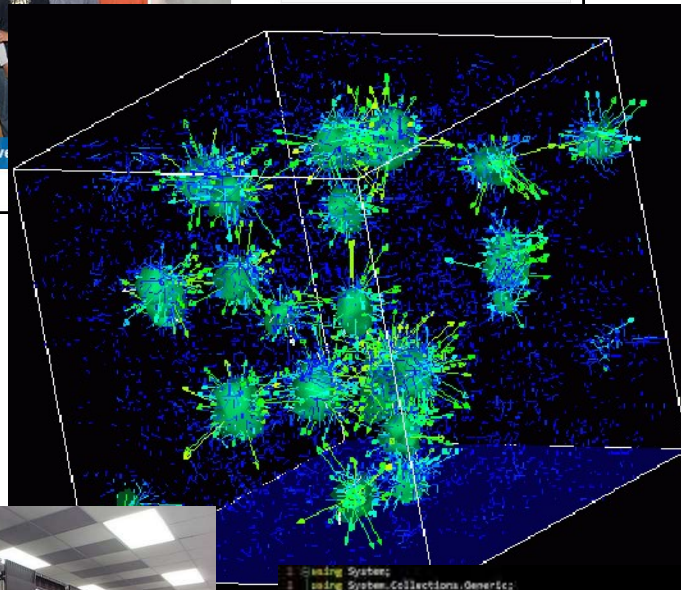
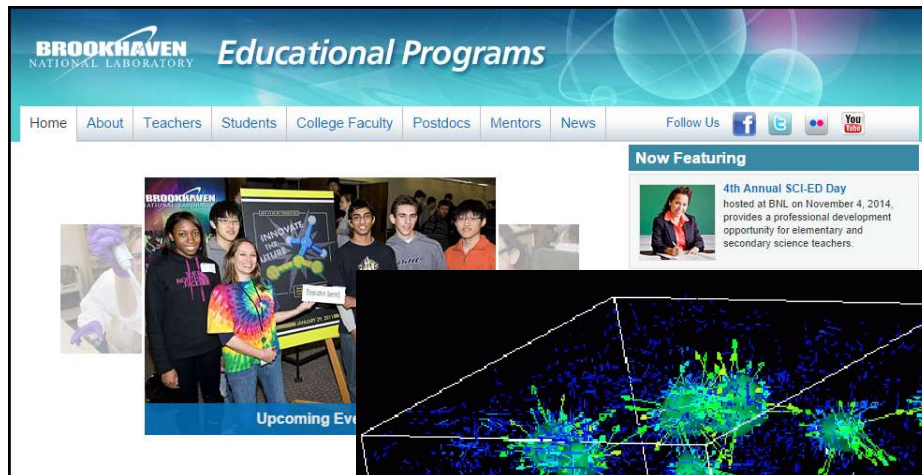
```
//private static double actual = .5;  
private static double actual = Math.PI / 4;
```



So now you know

- Numerical Integration attempts to find **the area under the curve**
 - It uses the sum of the areas of **successively smaller** and smaller strips
 - The strips can be sized according to the **Trapezoid**, **Midpoint**, or **Simpson's rule**
 - Simpson's method is the more accurate due to using parabolas
- Monte Carlo integration uses **random sampling**
 - The method calculates the ratio of the points below the curve to the total number of points – **the final ratio is the integral**
 - It requires millions/billions of samples to provide a few decimal points in accuracy
 - It may be the *only way* to take the integral of a very complex function

$$\begin{aligned}
\mathbf{F}^{(n)} = & \frac{\mu}{8\pi} \int_{r_1}^{r_2} \int_{s_1}^{s_2} \int_{y_1}^{y_2} \left(\frac{2}{R_a^3} + \frac{3a^2}{R_a^5} \right) \{ (\mathbf{R} \times \mathbf{b}) (\mathbf{t} \cdot \mathbf{n}) + \mathbf{t} [(\mathbf{R} \times \mathbf{b}) \cdot \mathbf{n}] \} \\
& \times \left(\frac{r - r_1}{r_2 - r_1} \frac{s - s_1}{s_2 - s_1} \right) ds dr dy \\
& - \frac{\mu}{4\pi (1 - \nu)} \int_{r_1}^{r_2} \int_{s_1}^{s_2} \int_{y_1}^{y_2} \left(\frac{1}{R_a^3} + \frac{3a^2}{R_a^5} \right) [(\mathbf{R} \times \mathbf{b}) \cdot \mathbf{t}] \mathbf{n} \left(\frac{r - r_1}{r_2 - r_1} \frac{s - s_1}{s_2 - s_1} \right) ds dr dy \\
& + \frac{\mu}{4\pi (1 - \nu)} \int_{r_1}^{r_2} \int_{s_1}^{s_2} \int_{y_1}^{y_2} \frac{1}{R_a^3} \{ (\mathbf{b} \times \mathbf{t}) (\mathbf{R} \cdot \mathbf{n}) + \mathbf{R} [(\mathbf{b} \times \mathbf{t}) \cdot \mathbf{n}] \} \left(\frac{r - r_1}{r_2 - r_1} \frac{s - s_1}{s_2 - s_1} \right) ds dr dy \\
& - \frac{\mu}{4\pi (1 - \nu)} \int_{r_1}^{r_2} \int_{s_1}^{s_2} \int_{y_1}^{y_2} \frac{3}{R_a^5} [(\mathbf{R} \times \mathbf{b}) \cdot \mathbf{t}] (\mathbf{R} \cdot \mathbf{n}) \mathbf{R} \left(\frac{r - r_1}{r_2 - r_1} \frac{s - s_1}{s_2 - s_1} \right) ds dr dy.
\end{aligned}$$



```

1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Windows.Forms;
9
10 namespace SimpleEvents
11 {
12     public partial class Form1 : Form
13     {
14         Person person = new Person();
15
16         public Form1()
17         {
18             InitializeComponent();
19             person.FirstName = "Christian";
20             person.LastName = "Foss";
21         }
22
23         private void button1_Click(object sender, EventArgs e)
24         {
25             person.HairColor = textBox1.Text;
26         }
27     }
28 }

```

Introducing Scientific Computing

Part 9 The Right Way to Shuffle

Goals

- Determine which is a more effective method to shuffle a deck of cards – **wash** or **ripple** shuffle
- Develop a statistic to **measure the randomness** of a card deck shuffle
- Understand what makes a shuffle approach good or bad in terms of producing randomized hands after initially starting from a deck in **perfect order**

Purpose: People who can't "riffle" shuffle a deck of cards often spread out the cards on the table and stir them up in what is called a "wash" shuffle. Most people think a riffle shuffle is better than a wash shuffle.

Can we measure if one way to shuffle a deck of cards is "better" than another?

Hypothesis: I think the riffle shuffle is a better way to scramble a deck because a wash shuffle does not always mix all parts of the deck evenly together.

Materials: A standard deck of 52 playing cards, a calculator, a pencil, and some scratch paper. We used Microsoft Excel 2013 to make the graphs in this poster.

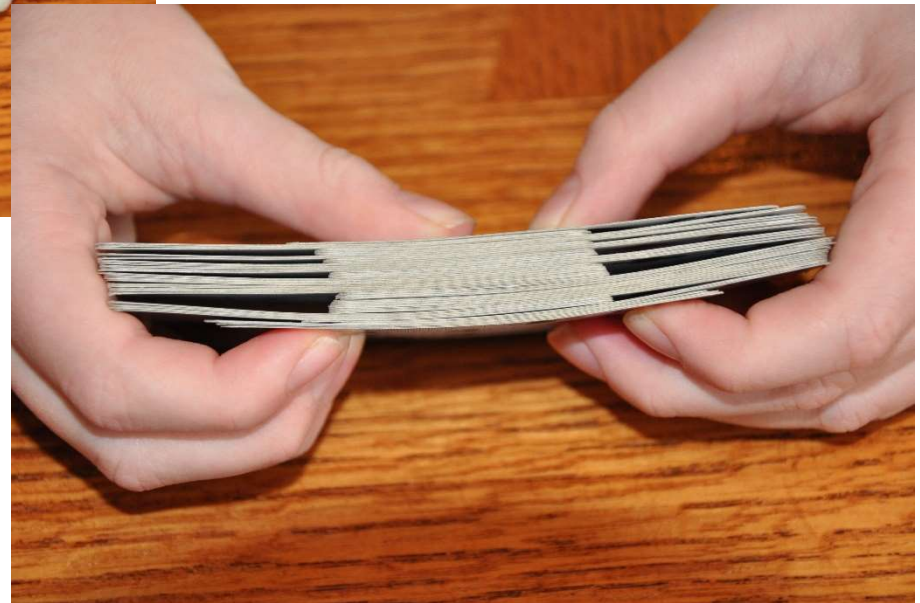
The “Wash” (or Toddler) Shuffle

Spread out all the cards on the table. Stir the pile randomly around several times, mixing the layers and separating any large clumps of cards.



The “Riffle” Shuffle

Cut the deck, then bend each half slightly upward from the middle, letting the corners interweave as you snap the halves back together.



Procedure

- Sort a 52-card deck first by suit, then by rank within each suit
- Starting each trial from this perfectly sorted deck “resets” the deck to give the same first deal every time
- Deal four hands clockwise with each player getting 13 cards
- Then **wash shuffle** the deck and determine the **randomness score**

Procedure

- Reset the deck again, **riffle shuffle** it, and find the new randomness score
- Then continue to riffle shuffle the same deck **nine** more times, each time recording the randomness score
- Write a computer program to perform **one million deals** and take the average randomness score each time

Scoring Deck Randomness

- In any card game, such as Poker, Rummy, Crazy-8s, or Spades, any deal that gives one player a lot of cards *with the same suit*, or *with the same rank*, or *with many partial runs*, is probably **not** a random deck
- It may not have been shuffled enough times after *the previous game*

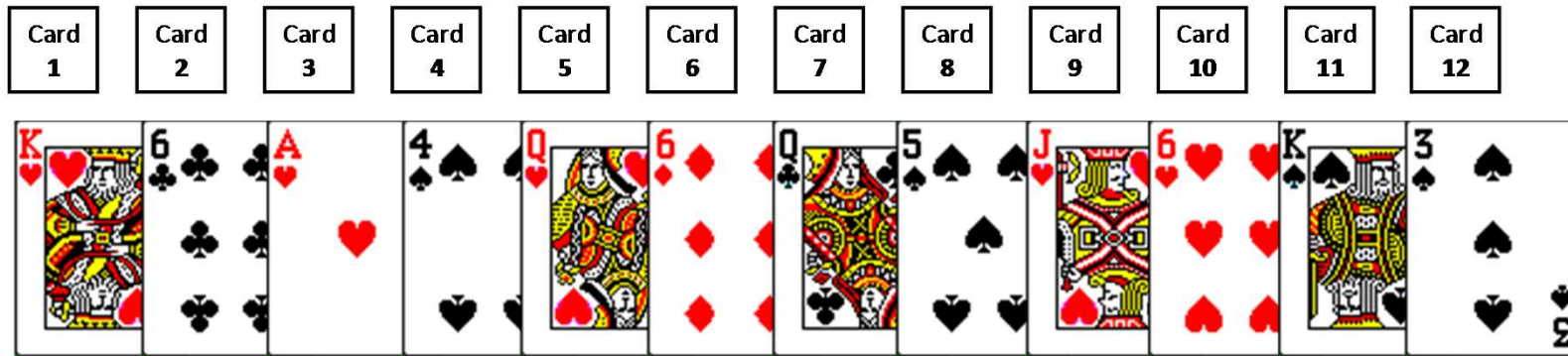
Lab (Open Discussion)

- If you start with a perfectly ordered deck, and shuffle it by some process before dealing it out to all players, then if the shuffle is good, the hands should be random
- If the hands are not very random, then the shuffle was not very effective
- **What metrics can we devise to measure the randomness of a deal and therefore determine which shuffle is better?**

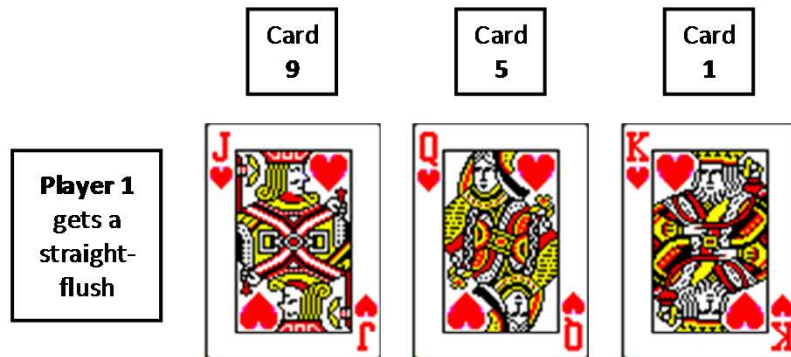
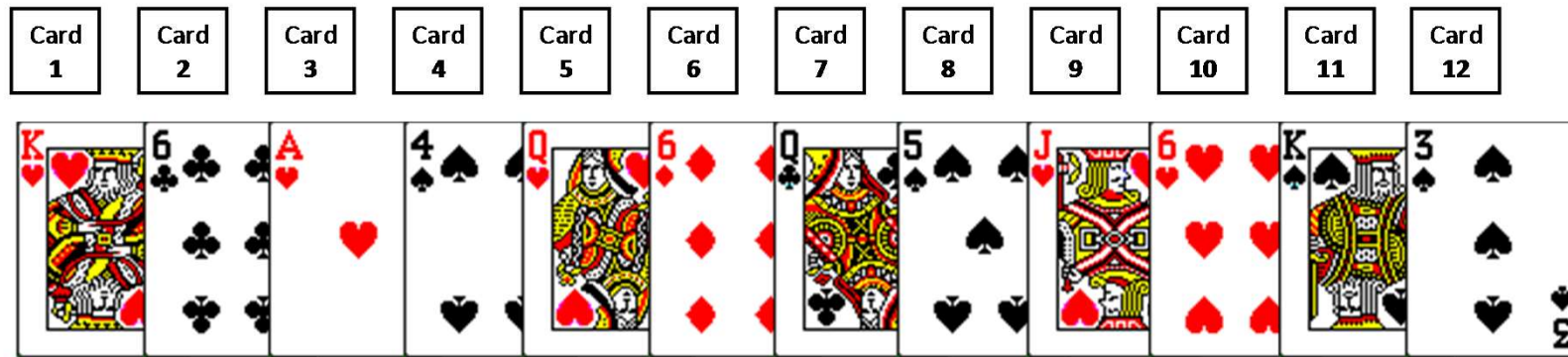
Deck Scoring

- To measure the randomness of a deck, count the cards in each hand using four different metrics and add up the total score
- A deck with a high score is not very random
 - For example, every hand in a four person deal should get about 3.25 cards of each suit
 - So after a deal, if **two** players each get **12 cards in a suit**, then that deck was probably not shuffled very well
- We have to analyze each **hand** individually ***after*** the deal is done to truly measure how good a shuffle process is at mixing up the cards

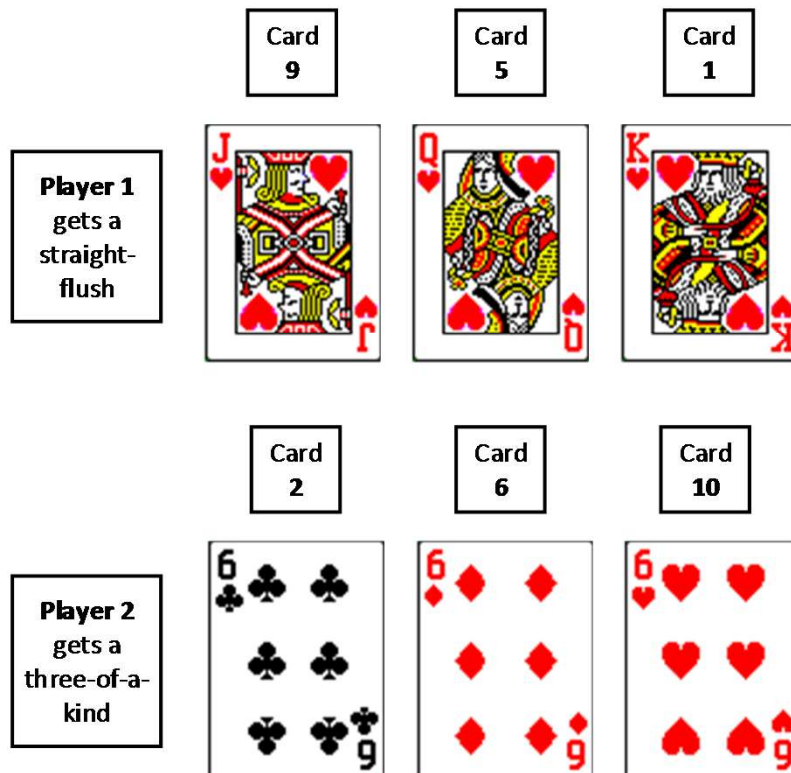
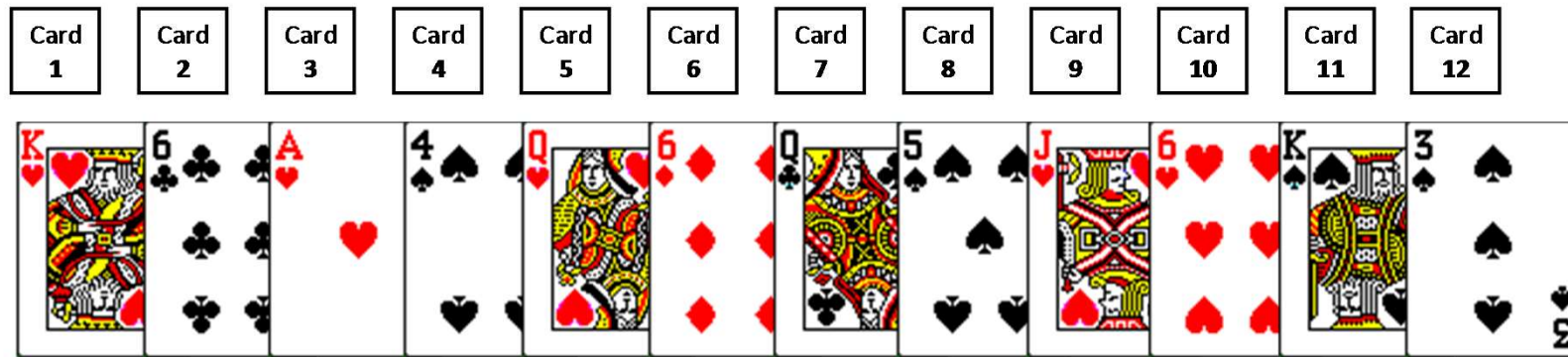
After a shuffle, the following cards in a deck seem like they are in a random order. But if they were dealt in order to four hands, each player would get a gift – so it is not the “next” card that determines a shuffle’s true randomness.



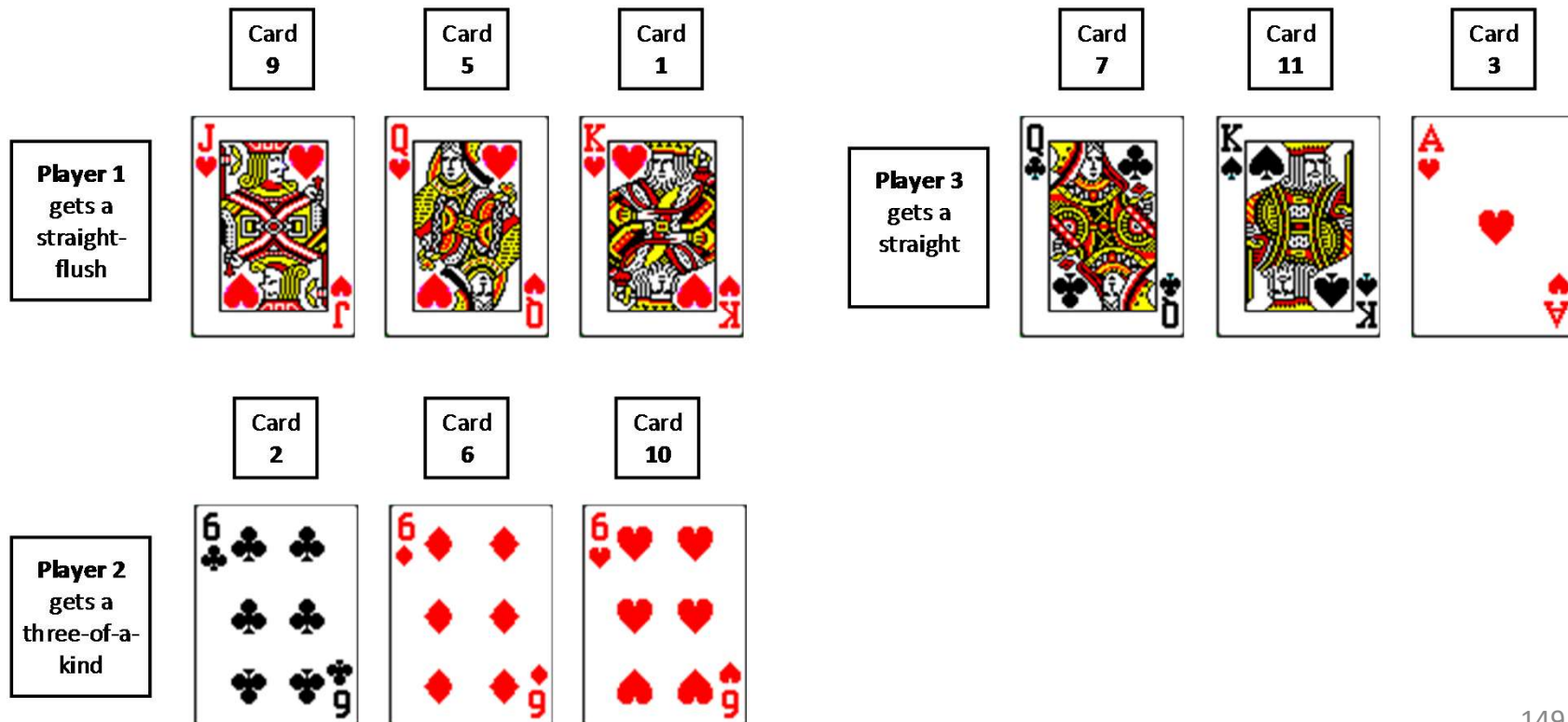
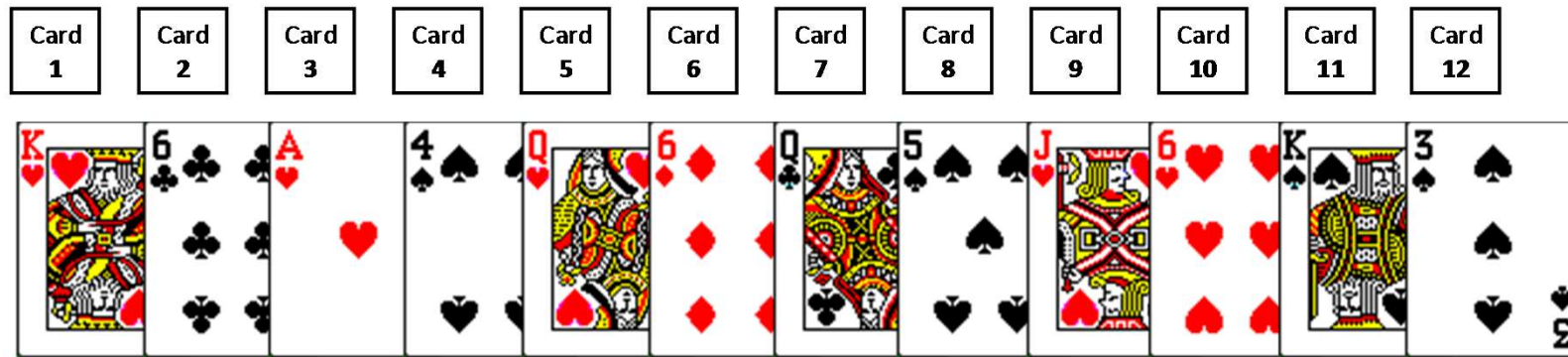
After a shuffle, the following cards in a deck seem like they are in a random order. But if they were dealt in order to four hands, each player would get a gift – so it is not the “next” card that determines a shuffle’s true randomness.



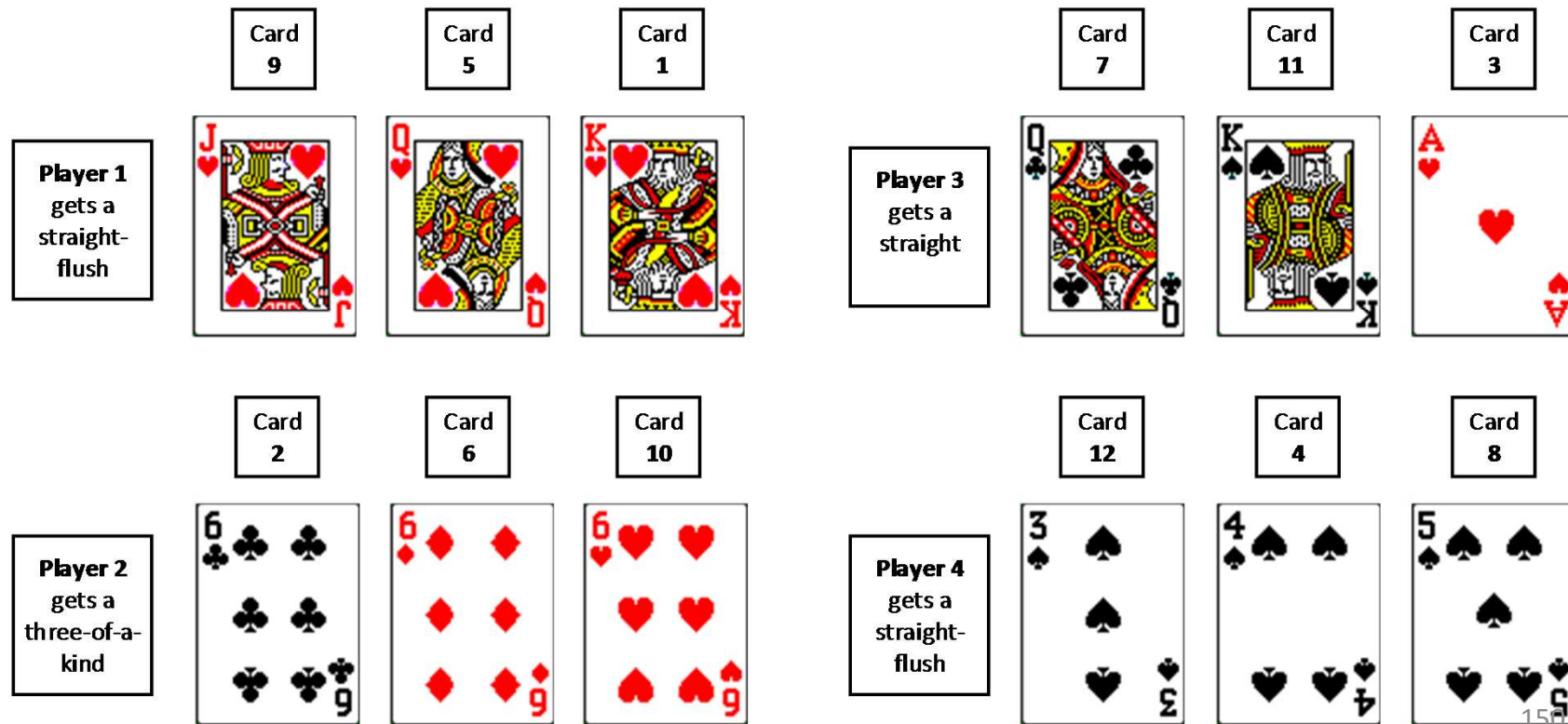
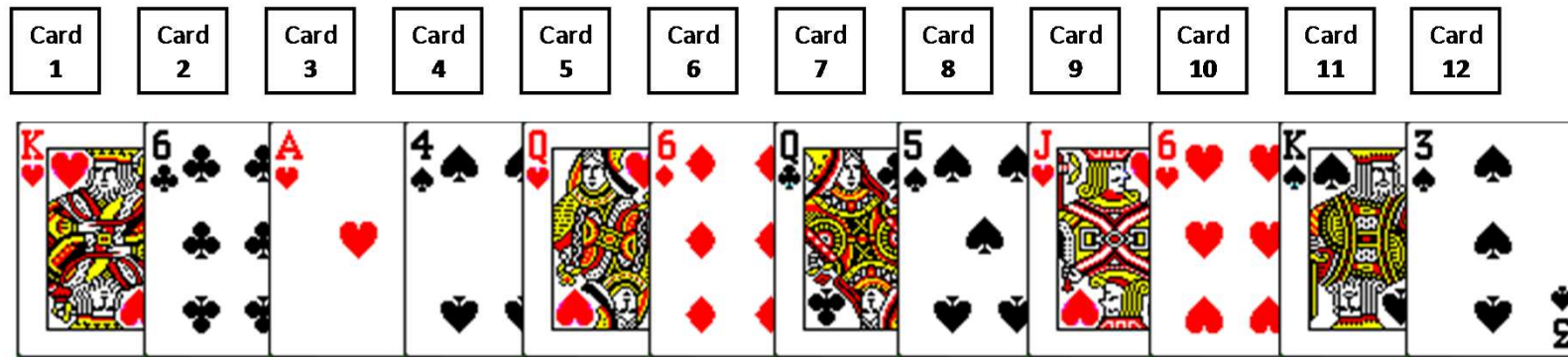
After a shuffle, the following cards in a deck seem like they are in a random order. But if they were dealt in order to four hands, each player would get a gift – so it is not the “next” card that determines a shuffle’s true randomness.



After a shuffle, the following cards in a deck seem like they are in a random order. But if they were dealt in order to four hands, each player would get a gift – so it is not the “next” card that determines a shuffle’s true randomness.



After a shuffle, the following cards in a deck seem like they are in a random order. But if they were dealt in order to four hands, each player would get a gift – so it is not the “next” card that determines a shuffle’s true randomness.



Deck Scoring – Distribution Points

- **Suit**

- In a good random deck, there should be on average about 3.25 cards in same suit per hand if dealing four hands ($13 / 4$)
- Score += $(\text{Observed} - \text{Expected})^2$

- **Rank**

- In a good random deck, there should be about one (1) of each rank in each hand (if dealing four hands)
- If you get a hand with four of a kind, the deck was probably not very random
- Score += $(\text{Observed} - \text{Expected})^2$

Deck Scoring – Distribution Points

- **Straight**

- In a good random deck, after dealing four hands, no hand should have runs of cards with consecutive (or nearly consecutive) increasing rank
- A run of **7-8-9-10** is less random than **7-8-9-Q** but neither are very likely
- **The average gap between successive ranks should be about equal to 4** if dealing four hands from a random deck
- $\text{Score} += (\text{Observed} - \text{Expected})^2$

Deck Scoring – Distribution Points

- **Straight Flush**

- Cards in same suit with increasing rank
- A run of cards with consecutive rank, all of the same suit, is **extremely unlikely** if the shuffle was good
- The score for having this appear in any hand after a deal should dramatically increase the randomness score
- Remember, the higher the randomness score, the worse the shuffle was (a higher score is considered a penalty)
- Score += (Observed-Expected)⁴

```

static double ScoreHand(Hand hand)
{
    double score = 0;

    // Add suit distribution points to score
    double avgCardsPerHand = (double)52 / numHands;
    double avgCardsPerSuitPerHand = avgCardsPerHand / (double)4;
    foreach (List<Card> suit in hand.suits)
        score += Math.Pow((suit.Count - avgCardsPerSuitPerHand), 2);

    // Add rank distribution points to score
    double avgSameRanksPerHand = (double)4 / numHands;
    foreach (List<Card> rank in hand.ranks)
        score += Math.Pow((rank.Count - avgSameRanksPerHand), 2);

    // Add "straight" distribution points
    double spacingStraight = (double)52 / avgCardsPerHand;
    List<Card> uniqueRanks = new List<Card>();
    foreach (List<Card> rank in hand.ranks)
        if (rank.Count > 0)
            uniqueRanks.Add(rank[0]);
    for (int rankNum = 0; rankNum < uniqueRanks.Count - 1; rankNum++)
    {
        int gapStraight = uniqueRanks[rankNum + 1].rank - uniqueRanks[rankNum].rank;
        if (gapStraight <= spacingStraight)
            score += Math.Pow((spacingStraight - gapStraight), 2);
    }

    // Add "straight flush" distribution points
    double spacingStraightFlush = (double)13 / avgCardsPerSuitPerHand;
    for (int suitNum = 0; suitNum < hand.suits.Length; suitNum++)
    {
        hand.SortSuitByRank(suitNum);
        for (int c = 0; c < hand.suits[suitNum].Count - 1; c++)
        {
            int gapStraightFlush = hand.suits[suitNum][c + 1].rank - hand.suits[suitNum][c].rank;
            if (gapStraightFlush <= spacingStraightFlush)
            {
                score += Math.Pow((spacingStraightFlush - gapStraightFlush), 4);
            }
        }
    }

    return score;
}

```

How do we simulate shuffling?

- The “**wash**” shuffle is essentially moving cards all around from their starting point
 - It is like the fast card dealer algorithm we studied in a prior class
- The “**riffle**” shuffle
 - Splits the deck in half
 - Takes a “chunk” of cards from the left half, then it overlays a chunk of cards from the right half, and so on
 - We found the number of cards in a chunk ranged randomly between 2 - 6 cards in actual practice

```

static void WashShuffle()
{
    for (int card = 0; card < 52; card++)
    {
        int swapCard = randomNumber.Next(52);
        int cardNumber = deck[card];
        deck[card] = deck[swapCard];
        deck[swapCard] = cardNumber;
    }
}

```

```

static void RiffleShuffle()
{
    int[] newDeck = new int[52];
    int leftPile = 0;
    int rightPile = 26;
    int card = 0;

    while (card < 52)
    {
        int takeLeft = randomNumber.Next(numCardsInShuffleChunk);
        for (int i = 0; (i < takeLeft) && (leftPile < 26) && (card < 52); i++)
        {
            newDeck[card] = deck[leftPile];
            leftPile++;
            card++;
        }
        int takeRight = randomNumber.Next(numCardsInShuffleChunk);
        for (int i = 0; (i < takeRight) && (rightPile < 52) && (card < 52); i++)
        {
            newDeck[card] = deck[rightPile];
            rightPile++;
            card++;
        }
    }
    deck = newDeck;
}

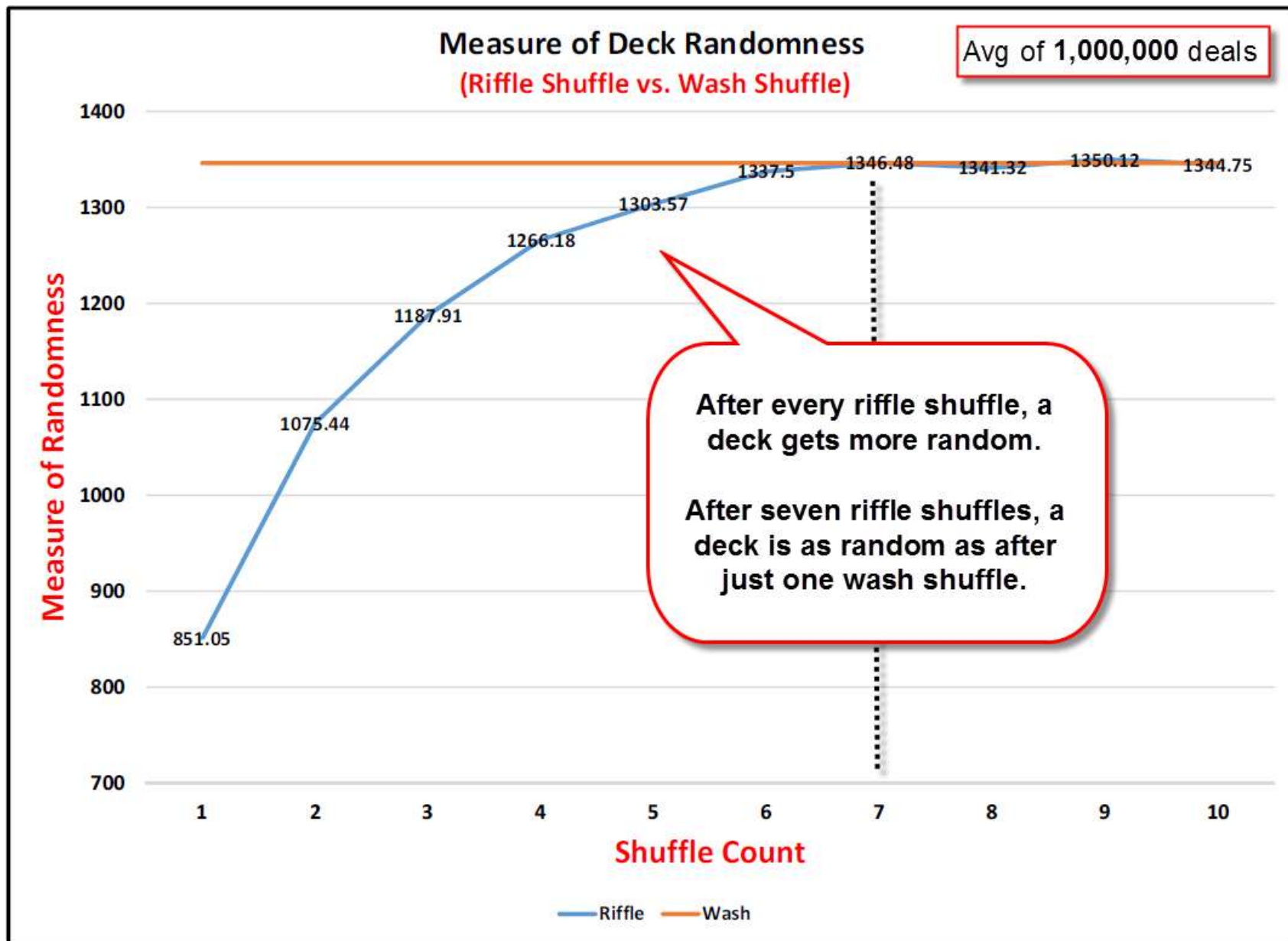
```

Measure of Randomness
Riffle Shuffle vs. Wash Shuffle

**Average of 1,000,000
computer generated deals
with 4 players (13 cards each)**

| Shuffle Type | Randomness |
|-------------------------|------------|
| After Wash Shuffles | 1,346.17 |
| After 1 Riffle Shuffle | 851.05 |
| After 2 Riffle Shuffle | 1,075.44 |
| After 3 Riffle Shuffle | 1,187.91 |
| After 4 Riffle Shuffle | 1,266.18 |
| After 5 Riffle Shuffle | 1,303.57 |
| After 6 Riffle Shuffle | 1,337.50 |
| After 7 Riffle Shuffle | 1,346.48 |
| After 8 Riffle Shuffle | 1,341.32 |
| After 9 Riffle Shuffle | 1,350.12 |
| After 10 Riffle Shuffle | 1,344.75 |

**It takes 7 riffle shuffles
to make a deal
as random as
just 1 wash shuffle!**



The “Riffle” Shuffle

Cut the deck, then bend each half slightly upward from the middle, letting the corners interweave as you snap the halves back together.



Analysis: My hypothesis about the riffle shuffle being better than the wash shuffle was **wrong**. A wash shuffle is much better than a riffle shuffle.

In fact, it took **seven riffle shuffles** to get the same randomness score as just a **single wash shuffle**.

Conclusion: Eventually the riffle shuffle does make a deck very random. But a big problem is that adults often only do 3 - 4 riffle shuffles between hands. So they might not know it, but their next deal is really not very random. The lesson learned is:

RIFFLE SHUFFLE SEVEN TIMES !!

And don't make fun of kids doing the wash shuffle! 😊

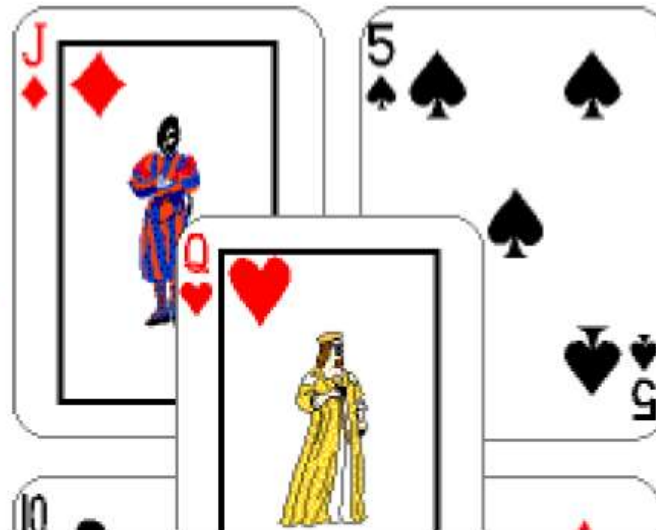
Disorder in the Deck

BY IVARS PETERSON 11:53AM, MARCH 18, 2003

SPONSOR MESSAGE

Card players sometimes get lazy, shuffling a deck fewer times than necessary to randomize the cards. Indeed, persistently sloppy shuffling can have a significant impact on play—an effect that experts (and gamblers) can exploit to their advantage.

However, the problem lay not in the computer but with human expectations. Subsequent research showed that hands in which suits are evenly distributed, that is, hands with four cards of one suit and three of each of the other suits, are actually more common in games in which people do the shuffling than they should be according to theory.



*

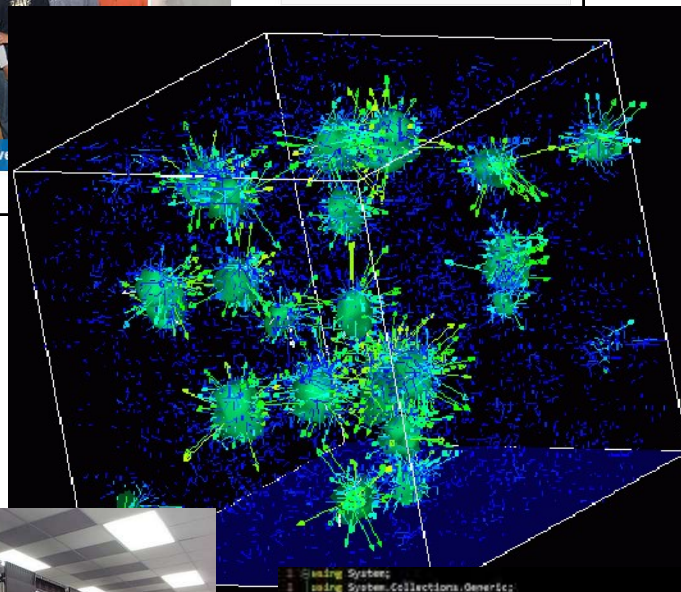
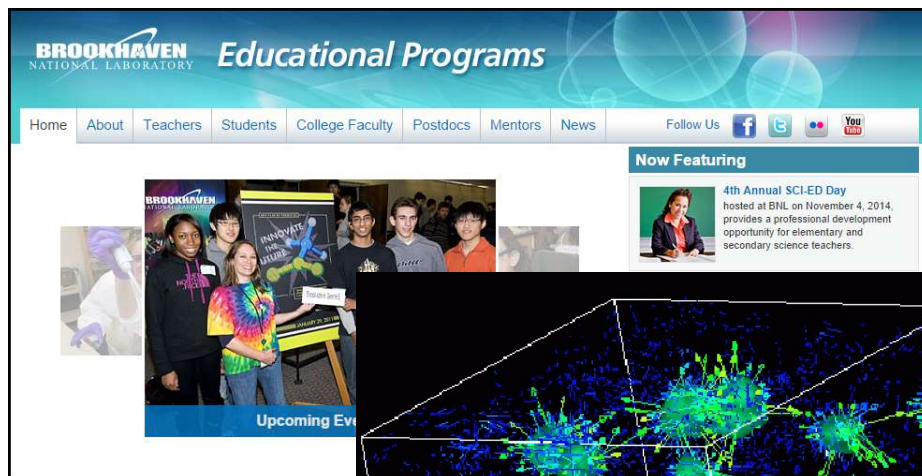
LAIR

We sh
the de
GSR-s
GSR-s

main result is a simple expression for the chance of any arrangement after any number of shuffles. This is used to give sharp bounds on the approach to randomness: $\frac{3}{2} \log_2 n + \theta$ shuffles are necessary and sufficient to mix up n cards.

Key ingredients are the analysis of a card trick and the determination of the idempotents of a natural commutative subalgebra in the symmetric group algebra.

ards. The



```

1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Windows.Forms;
9
10 namespace SimpleEvents
11 {
12     public partial class Form1 : Form
13     {
14         Person person = new Person();
15
16         public Form1()
17         {
18             InitializeComponent();
19             person.FirstName = "Christian";
20             person.LastName = "Foss";
21         }
22
23         private void button1_Click(object sender, EventArgs e)
24         {
25             person.HairColor = textBox1.Text;
26         }
27     }
28 }

```

Introducing Scientific Computing

Part 10 BNL SciComp Education Initiatives

A National Challenge

Computer Science in US Secondary Schools

Computer science courses in K-12 education are fading from the national landscape at the very moment they are needed most. Introductory secondary school computer science courses have

decreased in number

Computer Science

Source: <http://www>

Despite its critical

U.S. schools. The

2,100 of them

21

so

Today, most U.S. schools do not offer academic CS courses. Schools offer courses on how to use technology, but not courses that cover the fundamental concepts and skills of computing. In fact, only 19 percent of computer science classes

If technology is the future, however, we are doing a

the last

With technology touching nearly all parts of everyday life, can we really afford a society where most people lack even a rudimentary understanding of the automated devices surrounding them?

In 2012, Time magazine reported that the only serious computing class available to most students is AP computer science, which focuses on Java programming. And even that course was offered at just 10 percent of American high schools.

A few obstacles stand in the way of making computer science a core subject for students in the U.S., says Code.org's Roxanne Emadi. For one, policymakers often confuse "digital literacy" courses that teach students how to use programs like Microsoft Word and PowerPoint with real computer science instruction. In addition, computer science gets crowded out by schools' focus on other core subjects.

Computer
ence,
ics) fields that
participation
n school
dy by the
s.

A National Challenge

The Washington Post

Education

High school students are all about computers but get little instruction in computer science

By Donna St. George

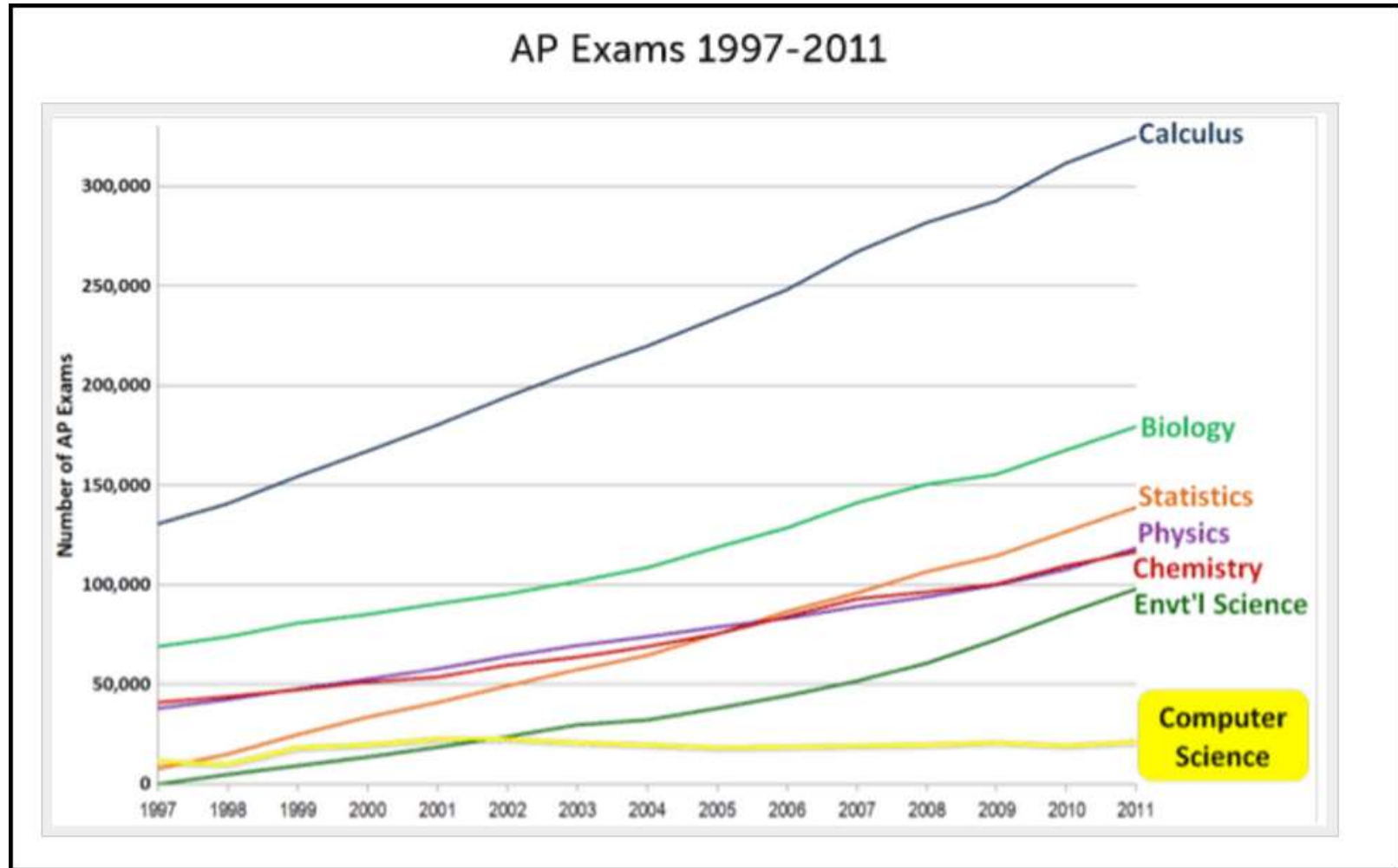
April 23, 2014

Their lives swirl in technology, but the nation's high school students spend little time studying the computer science that is the basis of it all. Few are taught to write lines of code, and few take classes that delve into the workings of the Internet or explain how to create an app.

In a world that went digital long ago, computer science is not a staple of U.S. education, and some schools do not even offer a course on the subject, including 10 of 27 high schools in Virginia's Fairfax County and six of 25 in Maryland's Montgomery County.

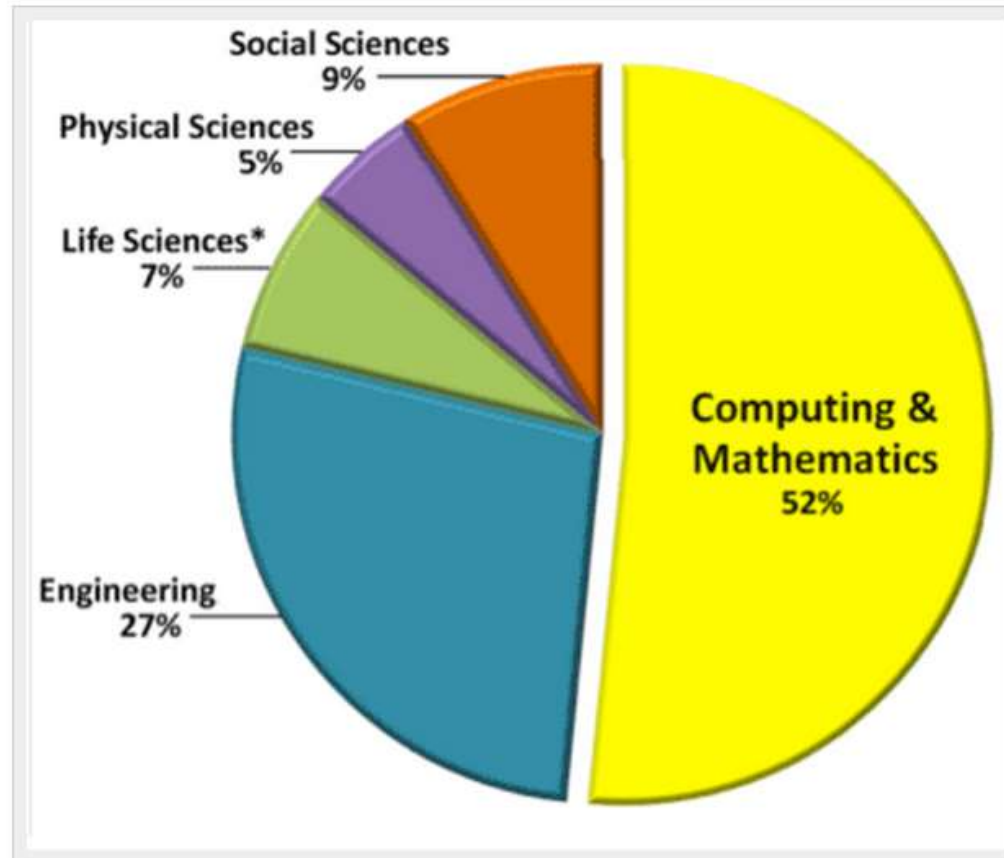
"It's shocking how little there is," said Rebecca Dovi, who has taught computer science for 17 years in Virginia schools and is an advocate for more courses statewide. Even when schools offer classes, she said, there are relatively few of them. "You might have one person teaching it in a school of 1,400 kids."

A National Challenge



The Irony

Projected Annual Growth of STEM Job Openings 2010-2020



Source: Jobs data are calculated from the Bureau of Labor Statistics (BLS), Employment Projections 2010-2020

The Challenge to BNL

- Scientists at BNL and other national laboratories have noted that the **startup latency** of interns is dramatically rising because young students do not possess foundational programming skills
- This latency means that for the ***initial 6-8 weeks of their assignment***, the interns are essentially unproductive as they must first acquire basic knowledge of how to write software
- Instead of learning key scientific principles from their mentor, the interns are spending their precious lab time **often working alone**, figuring out how to instruct the computer to perform rudimentary data processing tasks
- From the perspective of BNL scientists, the need to adeptly wield software tools has never been as urgent as it is today

The Reality

Kevin's Story

Modern-day scientists and engineers are spending more and more of their work days in front of the computer. As an example, consider my friend Kevin, who works in oceanography and mechanical engineering. Whoa, sounds like he's probably spending all day out on high-tech boats rigging together mechanical devices like MacGyver and collecting data from underwater sensors, right? This must be his typical work day -- hard hats and heavy-duty work gloves.



The Reality

Actually, Kevin spends less than 5% of his time out on the ocean; the other 95% of the time, he's sitting in front of the computer writing programs to clean up, transform, process, and extract insights from data collected out in the field. This is what Kevin looks like at work on most days:



The same story plays out for scientists and engineers in all sorts of fields: astronomers, biologists, physicists, aerospace engineers, economists, geneticists, ecologists, environmental engineers, neuroscientists ... the list goes on and on. Modern-day science and engineering is all about processing, analyzing, and extracting insights from data.

SciComp vs CompSci

Scientific Computing

- Probability and Statistics
- Simulation and Modelling
- Data Pipelining
- Data Visualization
- Storing and Analyzing Very Large Datasets
- Parallel & Distributed Algorithms
- Speed and Accuracy Paramount
- Functional and Interpreted Languages
- Open-Ended Problems with Unknown Solutions

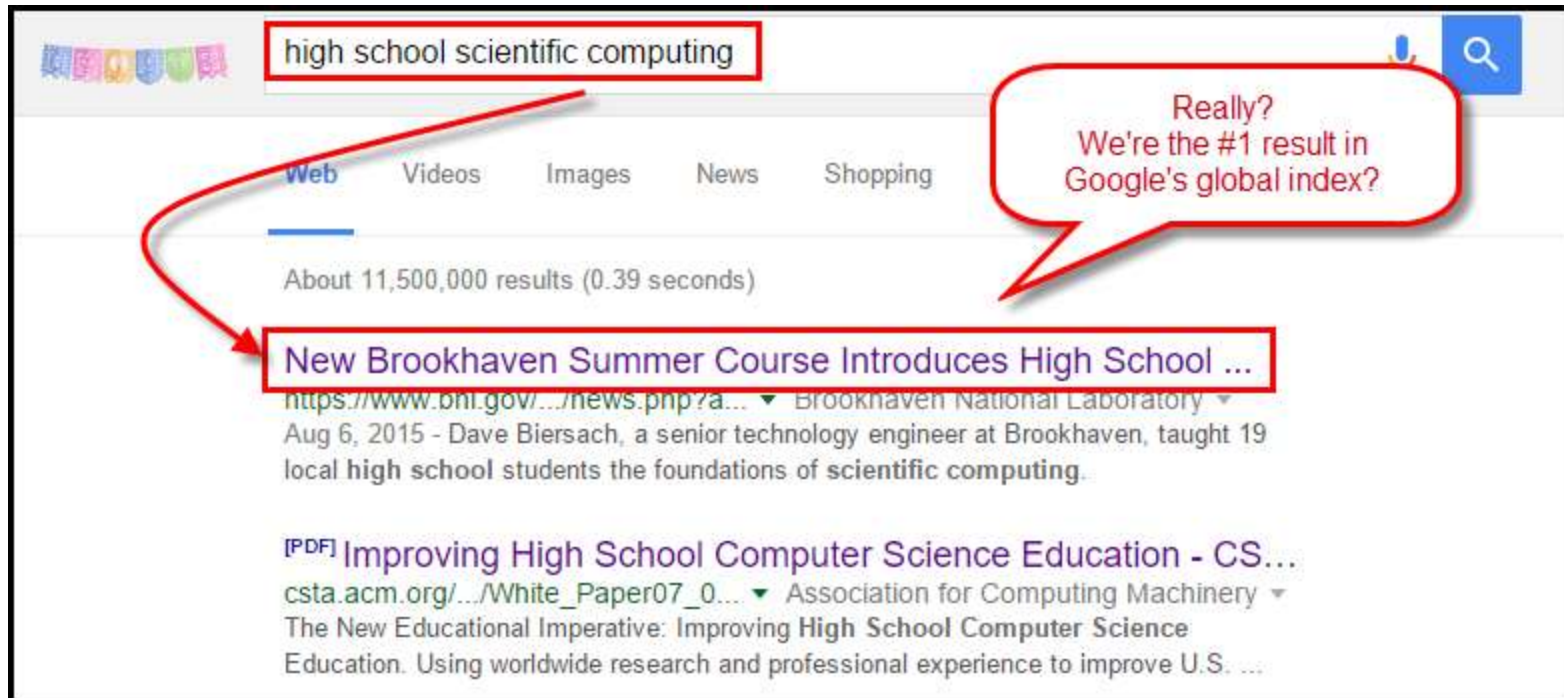
Computer Science

- General Purpose Data Structures
- Design Methodologies
- Procedural / Imperative Programming Languages
- Stand-Alone Programs
- Emphasis on Object-Orientation
- Simple Data Models
- Sequential Algorithms
- Less Graphics Intensive
- Directed Closed-Form Problems with Known Solutions

BNL Summer Workshops

- **SciComp 100 – Introduction to Scientific Computing**
 - No prior programming experience needed, Pre-Algebra desirable but not required
 - Ideal target for rising 6,7,8th grade students, one week onsite at BNL
- **SciComp 101 – Foundations of Scientific Computing**
 - No prior programming experience needed, Trigonometry desirable but not required
 - Ideal target for rising 9, 10, 11th grade students, two weeks onsite at BNL
- **SciComp 201 – Advanced Scientific Computing**
 - SciComp 101 prerequisite, Pre-Calculus required
 - Ideal target for rising 11, 12th grade students and HS graduates
 - Two weeks onsite at BNL
- **High School Summer Research Program (HSRP)**
 - Formal research projects supervised by BNL staff
 - Six to eight weeks onsite at BNL

BNL Summer Workshops



Dave Biersach, a senior technology engineer at Brookhaven, taught 19 local high school students the foundations of scientific computing.

[+ ENLARGE](#)

Last week, 19 Long Island high school students completed a two-week workshop designed to teach them the basics of computer programming for scientific research. With this new program, the U.S. Department of Energy's (DOE) Brookhaven National Laboratory looks to use its resources to begin to fill a gap in public science education while also building a pipeline to help identify and train the computer-literate researchers of tomorrow.

Scientific Computing Seminars

- BNL provides the cloud based compute resources and packaged learning materials for *weekly extracurricular* **SciComp Seminars**
 - These after-school seminars focus on getting students to produce working code to solve scientific problems. This is not a generic “computer” club
 - BNL researchers travel to Long Island schools on a rotating basis to sit with educators and students to collaborate and complete the sequence of prescribed hands-on exercises
- The local school educator supervising the club does not need to have any prior programming experience, nor does the school currently need to have any offering in computer programming
- Throughout the academic year BNL will host colloquia to bring together the SciComp clubs from across Long Island to share experiences and hear from industry and scientific luminaries

Scientific Computing Seminars

| <i>Islip</i> | <i>2:10-3:10</i> | <i>Sayville</i> | <i>3:00-4:00</i> | <i>Freeport</i> | <i>2:45-3:45</i> | <i>ESM</i> | | <i>11:00-12:00</i> | <i>WHB</i> | <i>3:10-4:10</i> | <i>Smithtown</i> | <i>3:00-4:00</i> |
|--------------|------------------|-----------------|------------------|-----------------|------------------|------------|-----|--------------------|------------|------------------|------------------|------------------|
| | Mon | | Tue | | Wed | | | Thu or Fri | | Thu | <i>East</i> | Fri |
| | | | | 1 | 7-Oct-15 | | | | | | | |
| | | | | | | 1 | FRI | 16-Oct-15 | | | | |
| 1 | 19-Oct-15 | 1 | 20-Oct-15 | | | | | | 1 | 22-Oct-15 | | |
| 2 | 26-Oct-15 | 2 | 27-Oct-15 | 1 | 28-Oct-15 | 2 | FRI | 30-Oct-15 | 2 | 29-Oct-15 | 1 | 30-Oct-15 |
| 3 | 2-Nov-15 | | | 2 | 4-Nov-15 | 3 | FRI | 6-Nov-15 | 3 | 5-Nov-15 | 2 | 6-Nov-15 |
| 4 | 9-Nov-15 | 3 | 10-Nov-15 | | | 4 | FRI | 13-Nov-15 | 4 | 12-Nov-15 | 3 | 13-Nov-15 |
| 5 | 16-Nov-15 | 4 | 17-Nov-15 | 3 | 18-Nov-15 | 5 | THU | 19-Nov-15 | 5 | 19-Nov-15 | 4 | 20-Nov-15 |
| 6 | 23-Nov-15 | | | | | | | | | | | |
| 7 | 30-Nov-15 | 5 | 1-Dec-15 | 4 | 2-Dec-15 | 6 | THU | 3-Dec-15 | 6 | 3-Dec-15 | 5 | 4-Dec-15 |
| 8 | 7-Dec-15 | 6 | 8-Dec-15 | 5 | 9-Dec-15 | 7 | FRI | 11-Dec-15 | 7 | 10-Dec-15 | 6 | 11-Dec-15 |
| 9 | 14-Dec-15 | 7 | 15-Dec-15 | 6 | 16-Dec-15 | 8 | THU | 17-Dec-15 | 8 | 17-Dec-15 | 7 | 18-Dec-15 |
| 10 | 11-Jan-16 | 8 | 12-Jan-16 | 7 | 13-Jan-16 | 9 | FRI | 15-Jan-16 | 9 | 14-Jan-16 | 8 | 15-Jan-16 |
| | | 9 | 19-Jan-16 | | | 10 | FRI | 22-Jan-16 | 10 | 21-Jan-16 | 9 | 22-Jan-16 |
| 11 | 25-Jan-16 | 10 | 26-Jan-15 | | | 11 | THU | 28-Jan-16 | 11 | 28-Jan-16 | 10 | 29-Jan-15 |
| 12 | 1-Feb-16 | 11 | 2-Feb-16 | 8 | 3-Feb-16 | 12 | FRI | 5-Feb-16 | 12 | 4-Feb-16 | 11 | 5-Feb-16 |
| 13 | 8-Feb-16 | 12 | 9-Feb-16 | 9 | 10-Feb-16 | 13 | THU | 11-Feb-16 | 13 | 11-Feb-16 | 12 | 12-Feb-16 |
| 14 | 22-Feb-16 | 13 | 23-Feb-16 | 10 | 24-Feb-16 | 14 | FRI | 26-Feb-16 | 14 | 25-Feb-16 | 13 | 26-Feb-16 |
| 15 | 29-Feb-16 | 14 | 1-Mar-16 | 11 | 2-Mar-16 | 15 | THU | 3-Mar-16 | 15 | 3-Mar-16 | 14 | 4-Mar-16 |
| 16 | 7-Mar-16 | 15 | 8-Mar-16 | 12 | 9-Mar-16 | 16 | FRI | 11-Mar-16 | 16 | 10-Mar-16 | 15 | 11-Mar-16 |
| 17 | 14-Mar-16 | 16 | 15-Mar-16 | 13 | 16-Mar-15 | 17 | THU | 17-Mar-15 | 17 | 17-Mar-16 | 16 | 18-Mar-16 |
| 18 | 21-Mar-16 | 17 | 22-Mar-16 | 14 | 23-Mar-16 | | | | | | | |
| | | 18 | 29-Mar-16 | 15 | 30-Mar-16 | 18 | FRI | 1-Apr-16 | 18 | 31-Mar-16 | 17 | 1-Apr-16 |
| 19 | 4-Apr-16 | 19 | 5-Apr-16 | 16 | 6-Apr-16 | 19 | THU | 7-Apr-16 | 19 | 7-Apr-16 | 18 | 8-Apr-16 |
| 20 | 11-Apr-16 | 20 | 12-Apr-16 | 17 | 13-Apr-16 | 20 | FRI | 15-Apr-16 | 20 | 14-Apr-16 | 19 | 15-Apr-16 |
| 21 | 18-Apr-16 | 21 | 19-Apr-16 | 18 | 20-Apr-16 | 21 | THU | 21-Apr-16 | 21 | 21-Apr-16 | 20 | 22-Apr-16 |
| 22 | 16-May-16 | 22 | 17-May-16 | 19 | 18-May-16 | | | | 22 | 19-May-16 | 21 | 20-May-16 |

Scientific Computing Seminars

| Seminar | Topics | Demo | Challenge Problem |
|---------|--|---|--|
| 1 | Using your Remote Development Environment | Hello World! | Hello <yourname>! |
| 2 | Data Types, Operators, Loops | Average of first N integers (both while and for) | WALKTHRU - Integer Factorial (for and while) |
| 3 | Conditionals, Functions, Stacks, Recursion | Recursive Greatest Common Factor (GCD) | Recursive Factorial |
| 4 | Strings, Arrays, Newton's Method | Newton's Method, Add & Multiple Large Integers | Average Large Integers |
| 5 | Modulus Operator, Random Numbers | Prime Number Sieve, Random Cups | WALKTHRU - Deal Card Deck |
| 6 | Matrices, Cramer's Rule, Ternary Encoding | Cramer's Rule (3x3), Ternary Encoding | Tic-Tac-Toe - finish HasWon() function |
| 7 | Console Input, Quadratic Factorization | Factor Input | Factor Quadratic Monomial |
| 8 | Sorting | Initialize Array, Print if Swap needed | Bubble Sort vs Quick Sort |
| 9 | ASCII, Streams, Histograms, Frequency Analysis | Character Histogram, Caesar Encrypt | Caesar Decrypt |
| 10 | 2D Graphics, Cartesian Coordinates | Hello Graphics!, Draw Parabola, Sine and Cosine Waves | DrawPolynomial |
| 11 | Polar Coordinates, Projectile Motion | Draw Circles | DrawOlympicRings |
| 12 | 3D Data Visualization, Contour Interpolation | Contour Map Interpolation (IDW) | Calculate Root Mean Square Deviation (RMSD) |
| 13 | Matrix Algebra, 2D Affine Transformations | Draw Jim IFS, Draw Fern IFS | Draw Your Own IFS |
| 14 | Numerical Integration | Trapezoid, Midpoint, Monte Carlo of $y=1-x$ [0,1] | Monte Carlo Integration of π |
| 15 | Normal Distribution, Chi Squared | Normal Distribution (Abramowitz & Stegun) | Create & Analyze Pachinko Normal Distribution |
| 16 | Continued Fractions | Standard CF Generator and Expansion | Determine Period for Standard CF for Irrationals |
| 17 | Cluster Analysis | Theory of k-means Clustering | WALKTHRU - k-Means Clustering |
| 18 | 2D Maze Encoding and Searching | Maze Encoding and Drawing | Kids design & encode their own 2D Maze |
| 19 | Longest Repeated Substring | Suffix Sort | Longest Matching Prefix |
| 20 | Measuring Randomness | Develop Metric for Randomness of Card Deck | Compare Wash vs. Riffle Shuffle |
| 21 | Scramble Squares Combinatorics | Orientations, Backtracking | Kids input their own scramble squares and solve |
| 22 | Stencils Combinatorics | Permutations, Combinations, Variations | IsValidStencil() - if else stack |

Educational Resources

- Scientific Computing Seminars in year 2016-2017
 - BNL will continue to come in the afternoons (or during science research classes during the normal school day) **only** to those schools who are willing to commit to send a teacher over the summer of 2017 to receive hands-on training on how to run the seminars on their own during the 2017-2018 school year
 - BNL is looking to collaborate with a progressive district to earn grant money to cover the teacher stipend for this 3 week on-site summer workshop for educators
- BNL & district developed curriculum materials
 - SciComp **Units** – groupings of several lessons provided to educators for them to tailor to their classes as desired
 - Full semester **Elective** course in Scientific Computing - aligned to AP Computer Science Principles Curriculum Framework and based upon recommendations from the BNL Science Advisory Committee
 - Individual Research **Projects** in Scientific Computing – directly aligned with current scientific programs at BNL

SciComp 101

Summer 2015

Selected slides from
presentations developed by
student participants

Bigram Analysis

- Using Bigram Analysis, we were able to determine that this text was written in German.

Cipher Text Bigram Frequency

file:///D:/DaveB/SciComp 101/Class 12 - ASCII, S

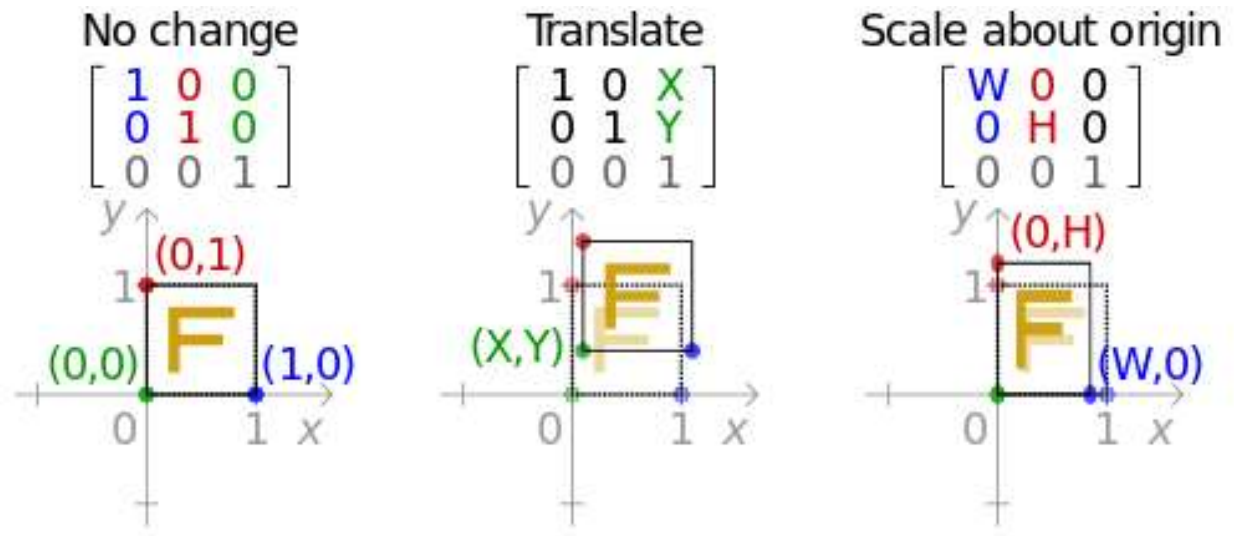
| ASCII | CHARS | FREQ |
|------------|-------|--------|
| (227, 228) | aa | 2.29 % |
| (227, 233) | aé | 2.10 % |
| (227, 239) | aï | 1.84 % |
| (228, 238) | âî | 1.60 % |
| (228, 239) | âï | 1.60 % |
| (233, 226) | éâ | 3.64 % |
| (235, 248) | éo | 1.05 % |
| (237, 239) | îï | 1.44 % |
| (238, 239) | îï | 1.71 % |
| (239, 227) | ïa | 2.49 % |
| (239, 228) | ïâ | 3.88 % |
| (239, 248) | ïo | 2.98 % |
| (239, 249) | ïù | 1.44 % |
| (248, 239) | oï | 1.13 % |
| (254, 239) | _ï | 2.03 % |

| German | Speech |
|--------|--------|
| ER | 3.90 |
| EN | 4.44 |
| CH | 2.36 |
| DE | 2.31 |
| EI | 1.98 |
| TE | 1.98 |
| IN | 1.71 |
| ND | 1.68 |
| IE | 1.48 |
| GE | 1.45 |
| ST | 1.21 |
| NE | 1.19 |

Top 5: 14.16

13.03

Math

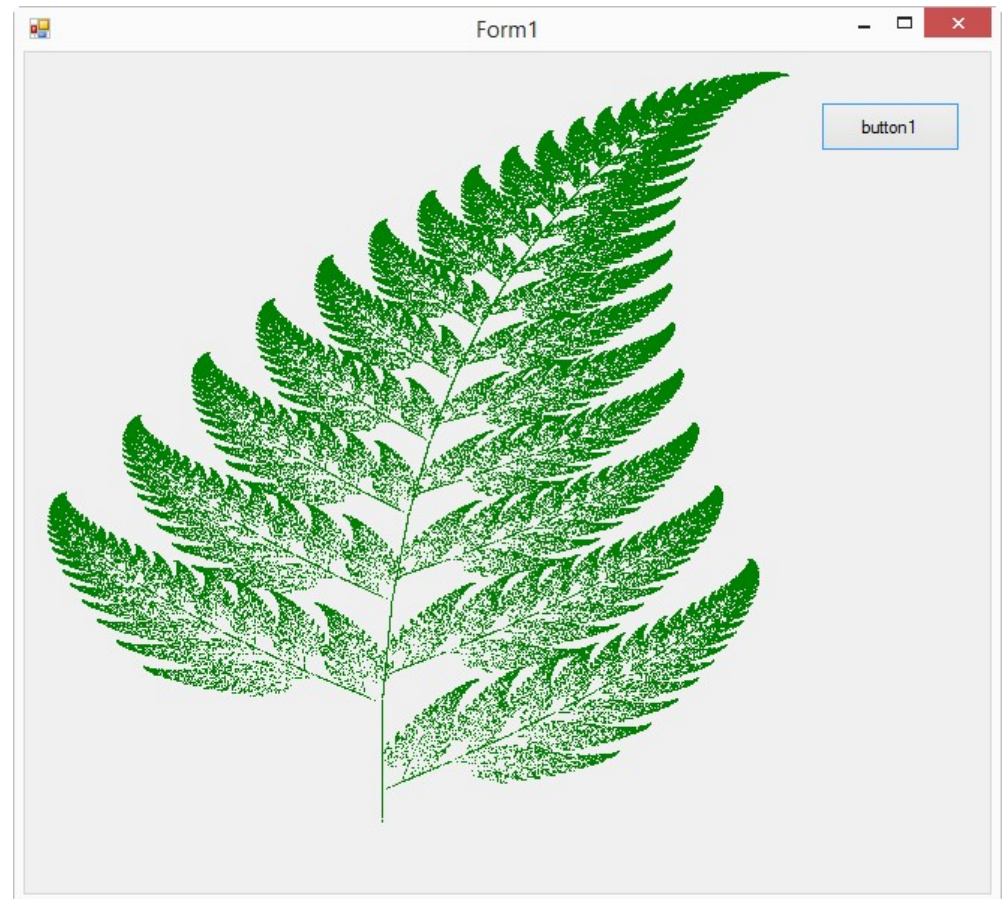
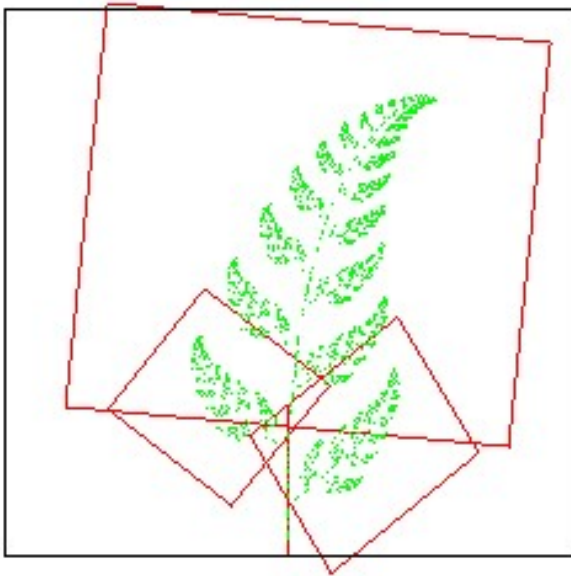


$$x'_n = ax_n + by_n + c$$

$$y'_n = dx_n + ey_n + d$$

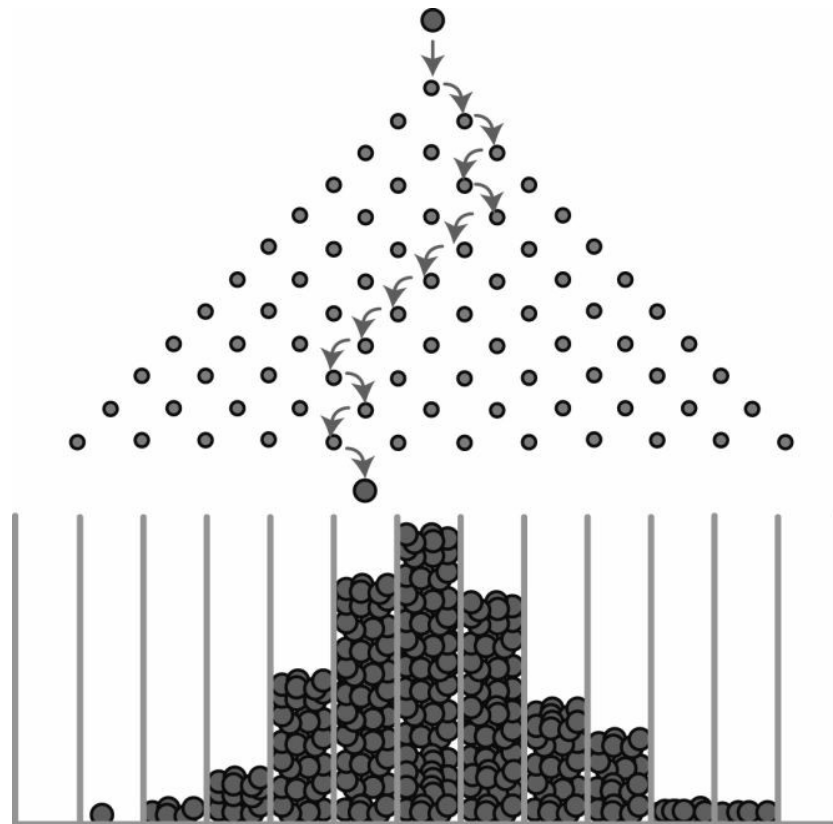
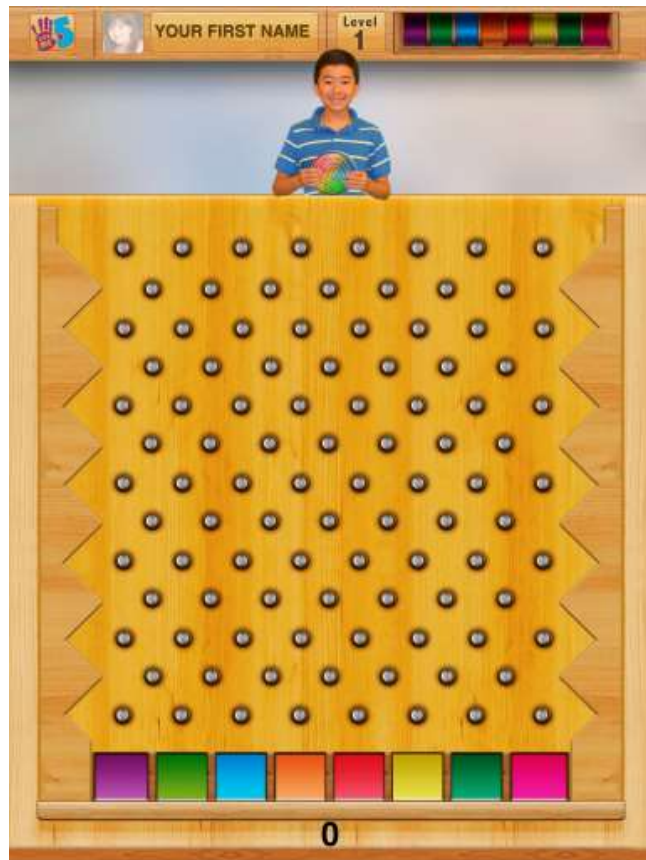
$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} = \begin{bmatrix} x'_1 & x'_2 & x'_3 \\ y'_1 & y'_2 & y'_3 \end{bmatrix} \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix}^{-1}$$

Results



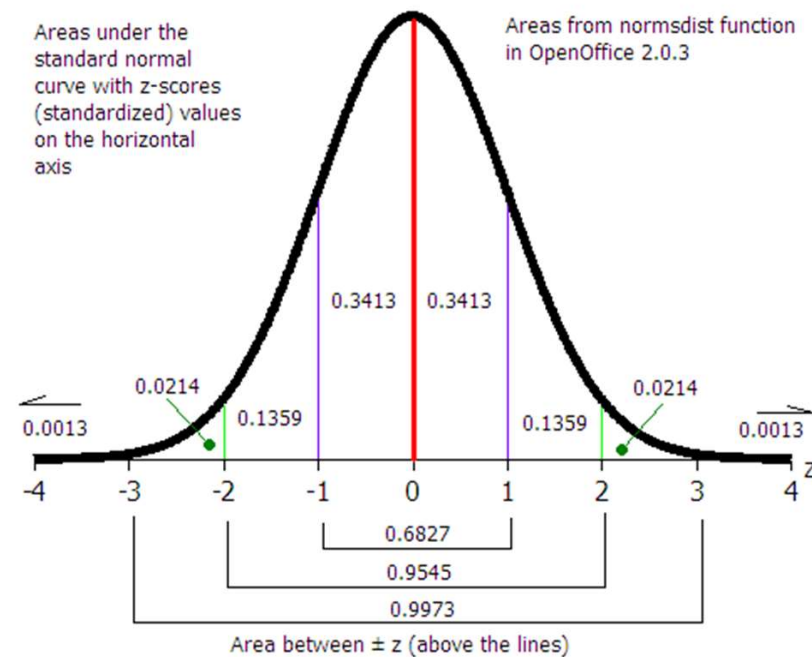
A Pachinko Distribution

We will be testing whether a Pachinko distribution can approximate a normal curve.

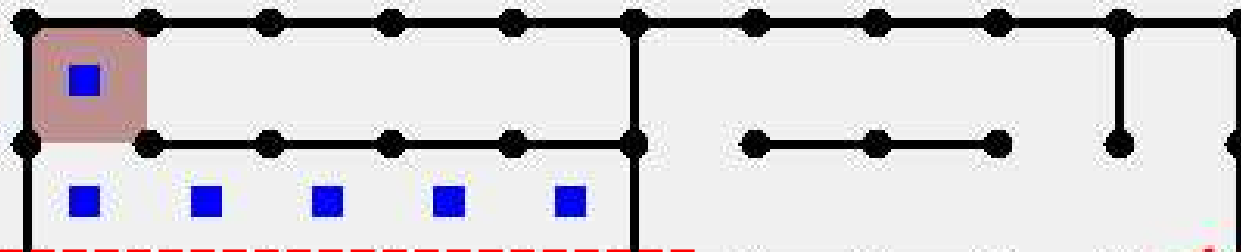


Results

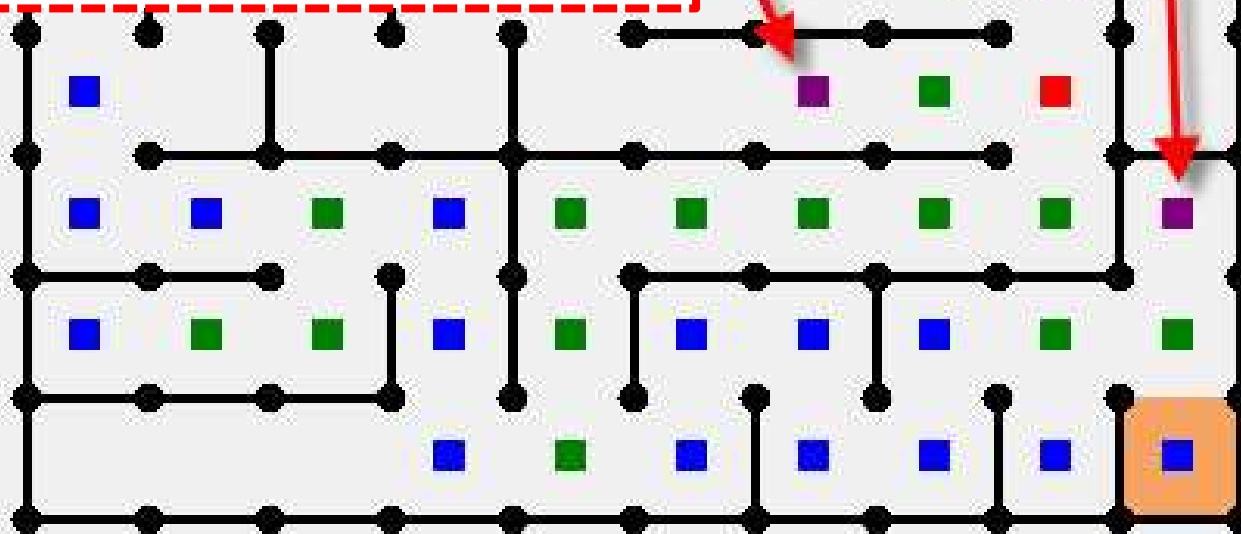
- The Pachinko distribution method does not approximate a bell curve.
- The Chi Square test suggests if the discrepancies between the observed and the expected values are statistically significant.



Maze Solver



Using an adjacency matrix reduces the number of steps averaging in a 46% improvement!



Depth First

☐ Wait for Step

☒ Limit Path

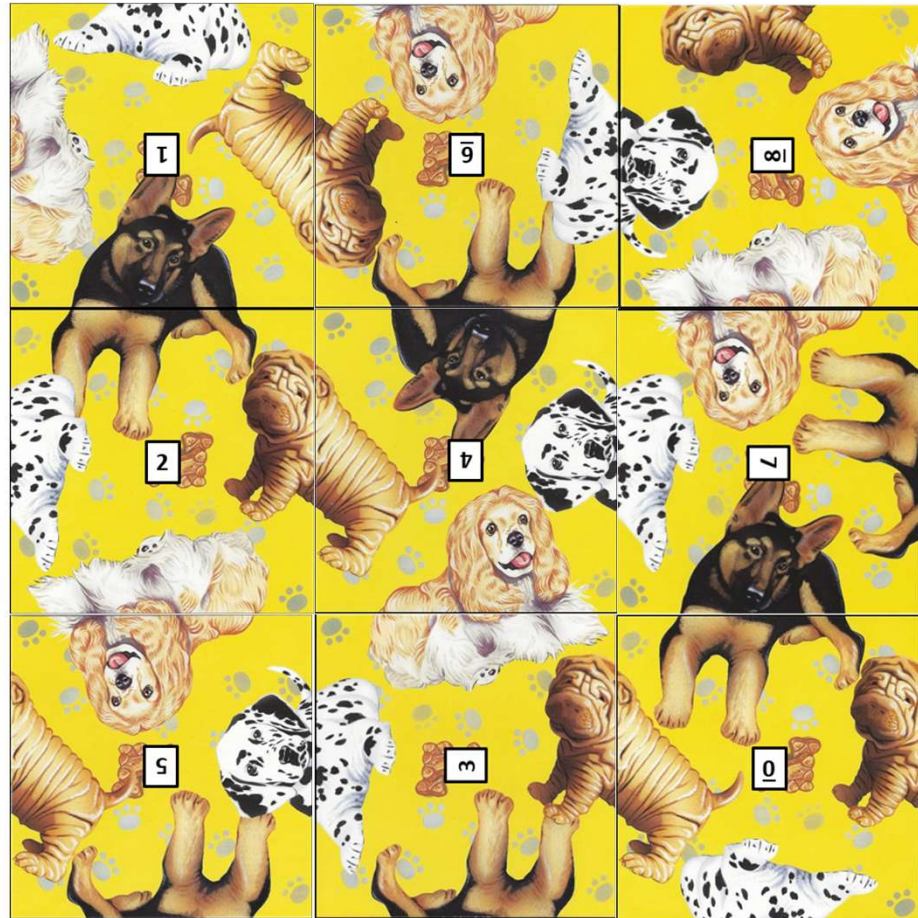
Next Step

Total Steps:

94

What are Scramble Squares?

- 9 tiles in a 3 x 3 matrix
- 4 full images cut into 4 half images per tile
- Each tile can be rotated in 4 different positions
- Inside edge of adjacent tiles must complete a full image (a complimentary “match”)
- Edges around outside of entire matrix do not need to match the other side
- There may be multiple correct layouts



Thank you!

- That you have chosen to spend today with BNL discussing Scientific Computing and the role it plays in the future of children pursuing STEM careers – we want to thank you!
- The BNL Office of Educational Programs (OEP) is here to help connect talented and passionate students with the many research opportunities on campus and across the Department of Energy complex
- OEP also provides connections for educators seeking to improve their science knowledge and teaching skills – take advantage of the extended network OEP represents
- We are the Department of Energy – not the Department of Education. Our focus is necessarily mission oriented – but we all cherish a bright future for our children and our country

Kennedy at Rice University, 1962

“We choose to go to the moon in this decade and do the other things, not because they are easy, but because they are hard, because that goal will serve to organize and measure *the best of our energies and skills*, because that challenge is one that we are willing to accept, one we are unwilling to postpone, and one which we intend to win.”

